The University of Jordan

Authorization Form

I, Tasneem Mahmoud abu kabeer, authorize the University of Jordan to supply copies of my Thesis/ Dissertation to libraries or establishments or individuals on request, according to the University of Jordan regulations.

Signature: 

Date: 27/4/2011.

الجامعة الأردنيـــة
كلية الدراسات العليا

التاريـــخ: / /

نموذج رقم (١٨)
اقرار والتزام بالمعايير الأخلاقية والأمانة العلمية وقوانين الجامعة الأردنية وأنظمتها وتعليماتها لطلبة الماجستير

أنا الطالب: نسنيم محمود الوليد الرقم الجامعي: (8080208)

تخصص: علم الحاسوب الكليـــة: تكنولوجيا المعلومات

عنوان الرسالة: Intelligence Medical decision Support System (MDSS) For medical management Diabetes type II

اعلن بأنني قد التزمت بقوانين الجامعة الأردنية وأنظمتها وتعليماتها وقراراتها السارية المفعول المتعلقة باعداد رسائل الماجستير عندما قمت شخصيا" باعداد رسالتي وذلك بما ينسجم مع الأمانة العلمية وكافة المعايير الأخلاقية المتعارف عليها في كتابة الرسائل العلمية. كما أنني أعلن بأن رسالتي هذه غير منقولة أو مستلة من رسائل أو كتب أو أبحاث أو أي منشورات علمية تم نشرها أو تخزينها في أي وسيلة اعلامية، وتأسيسا" على ما تقدم فانني أتحمل المسؤولية بأنواعها كافة فيما لو تبين غير ذلك بما فيه حق مجلس العمداء في الجامعة الأردنية بالغاء قرار منحي الدرجة العلمية التي حصلت عليها وسحب شهادة التخرج مني بعد صدورها دون أن يكون لي أي حق في التظلم أو الاعتراض أو الطعن بأي صورة كانت في القرار الصادر عن مجلس العمداء بهذا الصدد.

توقيع الطالب: ..... التاريخ: ٢٠١١ / ٤ / ٢٠

تعتمد كلية الدراسات العليا
هذه النسخة من الرسالة
التوقيع ..... التاريخ .....

INTELLIGENCE MEDICAL DECISION SUPPORT SYSTEM (MDSS) FOR MEDICAL MANAGEMENT DIABETES TYPE II

By
Tasneem Mahmoud Abu-Kabeer

Supervisor
Dr. Saleh Al-Sharaeh

Co-Supervisor
Dr. Ferial al hayajneh

This Thesis was submitted in Partial Fulfillment of the Requirements for the Master's Degree of Computer Science

Faculty of Graduate Studies
The University of Jordan

April, 2011

II

COMMITTEE DECISION

This Thesis /Dissertation (Intelligence Medical decision support system (MDSS) for medical management diabetes type II) was successfully Defended and Approved on 17/4/2011.

Examination Committee	Signature
Dr. Saleh Al-sharaeh (Supervisor) Associate Professor of Parallel Processing & High Performance Computing and Wireless Networks	
Dr. Ferial al hayajneh (Co-Supervisor) Associate Professor of Nursing	
Dr. Mohammad Alshraideh (Member) Assistant Professor of Software Testing Using Evolutionary Algorithms	
Dr.Imad Salah (Member) Associate Professor of Complex Systems and Networks	
Dr. Mohammad al abbadi (Member) Associate Professor of Multimedia, Image Processing (Mutah University)	

تعتمد كلية الدراسات العليا
هذه النسخة من الرسالة
التوقيع ..... التاريخ .....

DIDECATION

"To my beloved parents,
Mr. Mahmoud Abu-Kabeer and Mrs .Sobhia habash
And those who love me"

ACKNOWLEDGEMENTS

In the name of Allah, the Most Beneficent Most Merciful

All praise is due to **Allah**, the source of all knowledge and strength. I acknowledge His infinite mercy and grace in making this work a success. And may His peace and blessings be upon His final messenger **Muhammad**, a guidance and inspiration to our lives.

The successful completion of this work was made possible by a number of major contributions from different persons, to whom I wish to express my due gratitude:

Professor Saleh Al-Sharaiah, thesis supervisor, for the patience guidance, encouragement and advice he has provided throughout my time as his student. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly.

Dr. Mohammad Al-Shraideh for the patience guidance, encouragement and advice he has provided throughout my time as his student.

Dr. Mousa Al-Akhras for providing fresh ideas and directions in both areas of machine learning and neural networks.

Dr. Ferial al hyagneh, thesis Co-Advisors and **Nurse Sahar Jaffal,** for answering my questions about some rudiments medical concepts related to this work.

Dr. Stavroula Mougiakakou, for her help and guidance in neural network and in diabetes.

My parents, I would never have been able to pursue this task without their cooperation and understanding. I am eternally indebted to my mother for all her prayers, concern, love and understanding. Whatever knowledge or abilities I have

today have their roots in the love and teachings with which she has nurtured me. The value system ingrained by my father has driven me to always challenge myself and work hard towards my goals. I would like to especially thank **my brothers and sister** for their friendship and support.

All my friends, Nesreen Hamad, Elham Ahmad, Heba Saadeh ,Saeda Al Hindi, and others thanks all for their support and encouragement. Without them, I won't be able to get out from the bottle neck easily.

Tasneem .M. Abu-Kabeer

Amman 2011

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network.
CDSS	Clinical Decision Support System.
CDSSs	Clinical Decision Support Systems.
CI	Computational Intelligence.
GRNN	General Regression Neural Network.
Hardlim	hard-limit transfer function
JUH	Jordan University Hospital.
LF	Linear Function.
LM	Levenberg–Marquardt.
ML	Machine Learning.
MLP	Multilayer Perceptron.
MLPs	Multilayer Perceptrons.
NN	Neural Network
PID	Pima Indians Diabetes.
PNN	Probabilistic Neural Network.
RBF	Radial Basis Network.
Rprop	Resilient Back Propagation Algorithm.
SOM	Self Organizing Maps.
TANH	Hyperbolic Tangent Sigmoid Function.
UCI	University of California Irvine
WBCD	Wisconsin Breast Cancer Data set.

INTELLIGENCE MEDICAL DECISION SUPPORT SYSTEM (MDSS) FOR MEDICAL MANAGEMENT DIABETES TYPE II

By
Tasneem Mahmoud Abu-Kabeer

Supervisor
Dr. Saleh Al-Sharaeh

Co- Supervisor
Dr. Ferial al hayajneh

ABSTRACT

Diabetes is the most common endocrine disease in all populations and all age groups. According to the World Health Organization, it affects around 194 million people worldwide, and this number is expected to increase to at least 300 million by 2025.

The only way for the diabetes patient to live with this disease is to keep the blood sugar as normal as possible without serious high or low blood sugars this is achieved when the patient uses a correct management (therapy) which may include diet and exercising, taking oral diabetes medication or using some form of insulin. On the other hand treating the diabetes disease is also a difficult, an expensive and a complex task for the medical staff. There are number of important things to record about the patient and disease that help the doctors to make an optimal decision about the patient to make his/her life better.

In the last decades, several tools and various methodologies have been proposed by the researchers for developing effective clinical decision support system (CDSS) in order to improve the ability of the physicians, one of these methodologies is artificial neural networks (ANN) that are computer paradigms that belong to the computational intelligence family and are inspired by the biological neural system.

.

In this thesis, a multilayer perceptron (MLP) feed forward neural network is used to develop a clinical decision support system (CDSS) for determine the regimen type of diabetes management. The input layer of the system includes 25 input variables; the output layer contains one neuron that will produce a number that represents the treatment regimen. The number of hidden layers and nodes in the hidden layers are determined through a cascade learning process. In the system, the missing data of a patient are handled using the substituting mean method. Furthermore, a resilient back propagation (Rprop) algorithm is used to train the

system. In particular, a 10-fold cross validation scheme was used to access the generalization of the proposed system. We performed experiments with the proposed system. We obtained 88.5% classification accuracy from the experiments made on data taken from 228 patient medical records suffering from diabetes (type II) -this kind of diabetes usually develops in overweight, older adults- these records collected from the Jordan University Hospital (JUH)

Chapter One
1. ITNRODUCTION

A major class of problems in medical science involves either disease diagnosis or disease treatment, based upon various symptoms, tests and previous state performed upon the patient. When several symptoms and tests are involved, the ultimate decision may be difficult to obtain, even for a medical expert. Consequently over the past few decades, this has given rise to computerized medical tools, proposed to aid the physician in making sense of the confusing data in existence of imprecision and uncertainty (Kiyana and Yildirim, 2003), these computerized tools are called clinical decision support systems (CDSSs) which are type of computer programs that can be developed through several kind of machine learning (ML) techniques and used to assist clinicians at the point of care (Yan et al., 2006).

In this chapter; section 1.1 illustrates a general overview of ML, section 1.2 concerns about CDSSs, section 1.3 gives a general overview of Diabetes mellitus and Diabetes Management, section 1.4 introduces the problem statement and thesis motivation, and thesis organization is illustrated in section 1.5.

1.1 Machine Learning Overview

ML studies computer algorithms for learning to do stuff, learning to complete a task, or to make accurate predictions, or to behave intelligently. The learning that is being done is always based on some sort of observations or data such as examples (the case in this thesis) and direct experience or instructions. So in general, machine learning is about learning to do better in the future based on what was experienced in the past (Mitchell, 1997).

Learning, like intelligence, covers such a broad range of processes that it is difficult to define precisely. A dictionary definition includes phrases such as "to gain knowledge, or understanding of, or skill in, by study, instruction, or experience," and "modification of a behavioral tendency by experience." Zoologists and psychologists study learning in animals and humans (Mitchell, 1997). There are several parallels between animals and machine learning. Certainly, many techniques in machine learning were derived from the efforts of psychologists to make more precise their theories of animal and human learning through computational models. It seems likely also that the concepts and techniques being explored by researchers in machine learning may illuminate certain aspects of biological learning(Fausett, 2001).

As regards machines, we might say, very broadly, that a machine learns whenever it changes its structure, program, or data (based on its inputs or in response to external information) in such a manner that it is expected future performance improves. Some of these changes, such as the addition of a record to a data base, fall comfortably within the area of other disciplines and are not necessarily better understood for being called learning. But, for example, when the performance of a speech-recognition machine improves after hearing several samples of a person's speech, we feel quite justified in that case saying that the machine has learned(Veropoulos, 2001).

One might ask "Why should machines have to learn?" There are several reasons why machine learning is important. Of course, we have already mentioned that the achievement of learning in machines might help us understand how animals and humans learn. But there are important engineering reasons as well(Mitchell, 1997). Some of these are:

- Some tasks cannot be defined well except by example; that is, we might be able to specify input/output pairs but not a concise relationship between inputs and desired outputs. We would like machines to be able to adjust their internal structure to produce correct outputs for a large number of sample inputs and thus suitably constrain their input/output function to approximate the relationship implicit in the examples(Fausett, 2001).
- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down.
- Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant re-design.
- New knowledge about tasks is constantly being discovered by humans. Continuing re-design of AI systems to conform to new knowledge is impractical, but machine learning methods might be able to track much of it.

The ML techniques have been employed by many disciplines to automate complex decision making and problem solving tasks. This field of research is immensely diverse, with applications in medicine, mathematics, computer science, chemistry, economics, business management and many other fields (Ramalingam et al., 2006).The majority of these techniques are concerned in pattern recognition problems. Pattern recognition is the act of taking in raw data and taking an action based on the "category" or the pattern (Duda et al., 2005). An example of pattern recognition problems is classification (Mitchell, 1997). In ML, the solution of pattern recognition problems lies within the field of supervised

learning (the base for this thesis). The task is to learn (induce) the relationship between the dependent attributes (input) and the designated attribute (output) from a set of examples, i.e. generalize information collected from given data to unseen data. A common supervised ML tools are: Multilayer Perceptron (MLP) neural network (Yan et al., 2006) and decision trees (Mitchell, 1997). Another category of problems in ML is unsupervised learning .unsupervised learning is a class of problems in which one seeks to determine how the data are organized. It is distinguished from supervised learning in that the learner is given only unlabeled examples (Geoffrey and Terrence,1999). One form of unsupervised learning is clustering which is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense among neural network models. The Self-organizing map **(SOM)** is an example of unsupervised ML tool (Geoffrey and Terrence,1999).

1.1.1 Application of Machine Learning

Because of their ability to identify patterns or trends in data, machine learning methods have been successfully applied in many industries review of (Fausett, 2001):

- **Spoken language understanding:** within the context of a limited domain, determine the meaning of something uttered by a speaker to the extent that it can be classified into one of a fixed set of categories.
- **Spam filtering:** identify email messages as spam or non-spam.
- **Credit card fraud detection:** identify credit card transactions (for instance) which may be fraudulent in nature.
- **Weather prediction:** predict, for instance, whether or not it will rain tomorrow.

- **Medical diagnosis:** diagnose a patient as a sufferer or non-sufferer of some diseases.

1.1.2 Machine Learning in the Medical Domain

Machine learning algorithms were from the very beginning designed and used to analyze medical data sets. Today machine learning provides several indispensable tools for intelligent data analysis. Especially in the last few years, the digital revolution provided relatively inexpensive and available means to collect and store the data. Modern hospitals are well equipped with monitoring and other data collection devices, and data is gathered and shared in large information systems (Veropoulos, 2001). Machine learning technology is currently well suited for analyzing medical data, and in particular there is a lot of work done in medical diagnosis in small specialized diagnostic problems. Data about correct diagnoses are often available in the form of medical records in specialized hospitals or their departments. All that has to be done is to input the patient records with known correct diagnosis into a computer program to run a learning algorithm (Fausett, 2001). This is of course an oversimplification, but in principle, the medical diagnostic knowledge can be automatically derived from the description of cases solved in the past. The derived classifier (usually called CDSSs) can then be used either to assist the physician when diagnosing new patients in order to improve the diagnostic speed, accuracy and/or reliability, or to train students or physicians non-specialists to diagnose patients in a special diagnostic problem (Mitchell, 1997).

1.2 Clinical Decision Support Systems Overview

In medical diagnosis, some of the scopes for ambiguity in the inputs are: the history (patient's description of the disease condition), physical examinations (especially in cases of uncooperative patients) and laboratory tests (faulty methods or equipment). Moreover, in the treatment, there are chances of drug reactions and specific allergies (Brause, 2001). All these factors may not be taken into account by a busy clinician attending hundreds of patients daily. Therefore, computers can help the clinician to reach an accurate diagnosis faster because they make fewer errors than humans in making analyses of complex data (Rasheed, 2009).

In the literature many researchers have given their definitions of CDSSs; Musen (1997) defined a CDSS as '' any piece of software that takes information about a clinical situation as inputs and that produces inferences as outputs that can assist practitioners in their decision making and that would be judged as "intelligent" by the program users'' . Miller and Geissbuhler (2007) defined a CDSS providing diagnostic decision support as a ''computer-based algorithm that assists a clinician with one or more component steps of the diagnostic process''. Sim as cited in (kong et al., 2008) defined CDSS as '' software that was designed to be a direct aid to clinical decision-making, in which the characteristics of an individual patient are matched to a computerized clinical knowledge base and patient specific assessments or recommendations are then presented to the clinician or the patient for a decision''. So in general CDSSs are computer programs designed to help healthcare professionals to make clinical decisions. In a sense, any computer system that deals with clinical data or knowledge is intended to provide decision support. These systems can be

used for diagnosis, treatment of health diseases, and future evaluation of the patient (Kong et al., 2008).

The implementation of these systems could have several beneficial effects; they improved patient safety by for example through reducing medication errors and adverse events; medical errors imply actions such as the conclusion of wrong diagnosis, the prescription of an incorrect treatment or the inefficient execution of correct treatment. Additionally, CDSSs improved clinical documentation and patient satisfaction. Moreover, they supported medical education and training and provided immediate feedback to patients (Thornett, 2001). Because of these benefits, CDSSs were applied in many medical domains, such as cardiology, gynecology, diabetes, etc (Veropoulos, 2001). In this thesis we will focus on the diabetes domain.

1.3 Diabetes mellitus and Diabetes Management Overview

Diabetes is the most common endocrine disease in all populations and all age groups. According to the World Health Organization, it affects around 194 million people worldwide, and that number is expected to increase to at least 300 million by 2025. Diabetes has become the fourth leading cause of death in developed countries and there is substantial evidence that it is reaching epidemic proportions in many developing and newly industrialized nations (Gan,2003), with evidence pointing to avoidable factors such as sedentary lifestyle and poor diet.

According to Americans Diabetes Association "diabetes is a disease in which the body does not produce or properly use insulin. Insulin is a hormone that is needed to convert sugar, starches and other food into energy needed for daily life. The cause of diabetes continues to

be a mystery, although both genetics and environmental factors such as obesity and lack of exercise appear to play role"(Americans Diabetes Association, 2002). There are two main types of diabetes mellitus: Type 1 Diabetes or Insulin-dependent diabetes mellitus usually seen in young people, but it can occur at any age [Guthrie and Guthrie, 2002].

Type 2 or Non insulin-dependent diabetes mellitus diabetes is the more common metabolic disorder that usually develops in overweight, older adults where the pancreas may produce adequate amounts of insulin to metabolize glucose (sugar), but the body is unable to utilize it efficiently. Over time, insulin production decreases and blood glucose levels rise. Patients with type 2 diabetes do not require insulin treatment to remain alive.

Diabetes like any other chronic diseases treatment and management is the most important thing to keep body function as much as possible to be normal to avoid the long-term complications include progressive development of the specific complications of retinopathy with potential blindness, nephropathy that may lead to renal failure, and/or neuropathy with risk of foot ulcers. Which make Treatment decision-making is complex and involves many factors.

In this thesis we mainly focused on diabetes type II treatment- as we said early this kind of diabetes may treated by oral hypoglycemic and/or insulin- which is associated with improved glycemic control and decreased diabetic complications. Further adjustments should be based on the A1C result, aiming for levels as close to the nondiabetic range as possible. Until glycemic goals are achieved, the patient should be seen at least monthly, and more frequently as indicated. An A1C should be obtained no more frequently than every three months. Medications should be adjusted according to blood glucose data.

The classes of medication used to treat diabetes patient with type two are summarized below:

- Metformin: is the only biguanide available in most of the world. It reduces hepatic glucose production in the presence of insulin. Metformin reduces A1C levels by 1–2%. In contrast to sulfonylureas, metformin is associated with weight loss or no weight gain, and has a lower risk of hypoglycemia. Unless contraindicated, metformin in combination with lifestyle changes is recommended as initial treatment for type 2 diabetes. Metformin can also be used in combination with insulin and sulfonylureas

- cs: stimulate insulin secretion and are most effective in treating non-obese, type 2 diabetics. The various Sulfonylureas have equivalent efficacy, reducing A1C by 1–2%. Sulfonylureas can be prescribed as monotherapy, or they can be combined with other oral agents or with insulin.

- Insulin: is the oldest of the currently available medications and, thereby, the one with the most clinical experience. Although initially developed to treat insulin-deficient type 1 diabetes, it has long been used to treat insulin-resistant type 2 diabetes. It is the most effective drug to decrease glycemia. In adequate doses, insulin can decrease any level of elevated A1C to meet a therapeutic goal.

1.4 Problem Statement and Thesis Motivation

There is a great responsibility lies on diabetic physicians to make a correct decision about what kind of treatment is suitable to each patient, this decision is important and at the same time it is difficult, important to patient since the only way for the patient to live with this

disease is to take the appropriate treatment, and difficult to physicians since it involves many factors.

So that the main aim of this thesis is to propose a clinical decision support system based on the patients' clinical records that were collected from JUH that can recognize the type of diabetes medication and support physician's decision. In order to accomplish this, a neural network with a resilient back propagation will be used. This thesis will also investigate the number of hidden neurons and hidden layers to observe the impact on the classification accuracy.

1.5 Thesis Organization

This thesis is organized as follows: preliminaries concepts and background information related to ANN and some literature review are presented in Chapter 2. Data preparation and system architecture are illustrated in Chapter 3. The experimental results of different architectures that were applied on the collected data sets are presented in Chapter 4. Chapter 5 concludes this research and proposes some future works.

Chapter two

2. Literature Review

2.1 Artificial Neural Networks Background

Computational Intelligence (CI) is a set of nature-inspired computational methodologies and approaches to address complex problems of the real world applications to which traditional methodologies and approaches are ineffective or infeasible (Thammano and Moolwong, 2009). One of the techniques that belong to the CI family is Artificial Neural Networks (ANNs) (usually called Neural Network (NN)) that were introduced in 1940s (further developed in 1980s) as an attempt to emulate the features of biological neural networks in order to address a range of difficult information processing, analysis and modeling problems (Fausett, 2001). They are a powerful data modeling tool that is able to capture and represent complex input/output relationships. Since the motivation for the development of NN technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain; as a result neural networks resemble the human brain in the following two ways:

1. A neural network acquires knowledge through learning.

2. A neural network's knowledge is stored within inter-neuron connection strengths known as weights.

ANNs have seen an explosion of interest over the last few years, and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics (Mehrotra et al., 1996). Indeed,

anywhere that there are problems of prediction or classification neural networks are being introduced (Haykin, 2001). This sweeping success can be attributed to a few key factors:

- **Power:** Neural network models can implicitly detect complex nonlinear (a term which is discussed in more details later in subsection 2.1.4.3) relationships between independent and dependent variables, also they have the ability to detect all possible interactions between predictor variables and be developed using multiple different training algorithms (Fausett, 2001).
- **Ease of use:** Neural networks learn by example. The neural network user gathers representative data, and then invokes training algorithms to automatically learn the structure of the data (Haykin, 2001). Although the user does need to have some heuristic knowledge of how to select and prepare data, how to select an appropriate neural network, and how to interpret the results,.

Neural networks are applicable in virtually every situation in which a relationship between the predictor variables (independents, inputs) and predicted variables (dependents, outputs) exists, even when that relationship is very complex and not easy to articulate in the usual terms of "correlations" or "differences between groups" (Fausett, 2001). A few representative examples of problems to which neural network analysis has been applied successfully are:

- **Credit assignment:** A variety of pieces of information are usually known about an applicant for a loan. For instance, the applicant's age, education, occupation, and many other facts may be available. After training a neural network on historical

data, neural network analysis can identify the most relevant characteristics and use those to classify applicants as good or bad credit risks (Haykin, 2001).

- **Monitoring the condition of machinery:** Neural networks can be instrumental in cutting costs by bringing additional expertise to scheduling the preventive maintenance of machines (Mehrotra et al., 1996). A neural network can be trained to distinguish between the sounds a machine makes when it is running normally ("false alarms") versus when it is on the verge of a problem. After this training period, the expertise of the network can be used to warn a technician of an upcoming breakdown, before it occurs and causes costly unforeseen "downtime".

Due to the importance of ANNs as discussed above, the next subsections will concern about giving a general background of the biological model that inspired the development of ANNs and comparing it with the basic model of an ANN. Also the activation functions that are commonly used when building an ANN model are also discussed. Beside the common two types of ANNs and the training algorithms, the designing issues that are related to ANNs are also presented.

2.1.1 The Biological Model

The basic model of the artificial neuron is founded upon the functionality of the biological neuron. By definition, “Neurons are basic signaling units of the nervous system of a living being in which each neuron is a discrete cell whose several processes are from its cell body”.Each neuron is a specialized cell which can propagate an electrochemical signal. The neuron has a branching input structure (the dendrites), a cell body, and a branching output structure (the axon). The axons of one cell connect to the dendrites of another via a synapse

as shown in Figure 2.1. When a neuron is activated, it fires an electrochemical signal along the axon. This signal crosses the synapses to other neurons, which may in turn fire. A neuron fires only if the total signal received at the cell body from the dendrites exceeds a certain level (the firing threshold). The strength of the signal received by a neuron (and therefore its chances of firing) critically depends on the efficacy of the synapses. Each synapse actually contains a gap, with neurotransmitter chemicals poised to transmit a signal across the gap junction (Haykin, 2001).

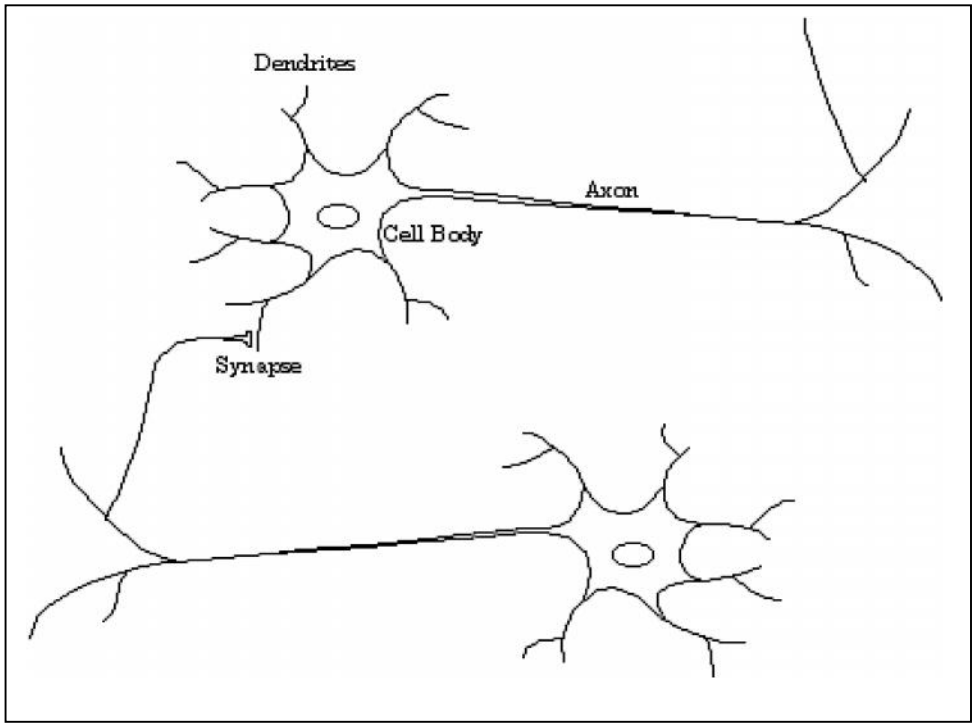

Figure 2. 1 Biological Neuron , (Haykin, 2001).

The brain which is principally composed of a very large number (10,000,000,000) of neurons, massively interconnected (with an average of several thousand interconnects per neuron, although this varies enormously) will analyze all patterns of signals sent, and from that information it interprets the type of information received. Thus, from a very large number of neurons (each performing a weighted sum of

its inputs, and then firing a binary signal if the total input exceeds a certain level) the brain manages to perform extremely complex tasks (Haykin, 2001). Of course, there is a great deal of complexity in the brain which has not been discussed here, but it is interesting that ANNs can achieve some remarkable results using a model not much more complex than this.

2.1.2 The Basic Model

The basic model of ANNs is an input layer consisting of nodes that simply accept the input values. The outputs of neurons in a layer are inputs to neurons in the next layer. The last layer is called the output layer. Layers between the input and output layers are known as hidden layers. Figure 2.2 illustrates a diagram for this architecture.

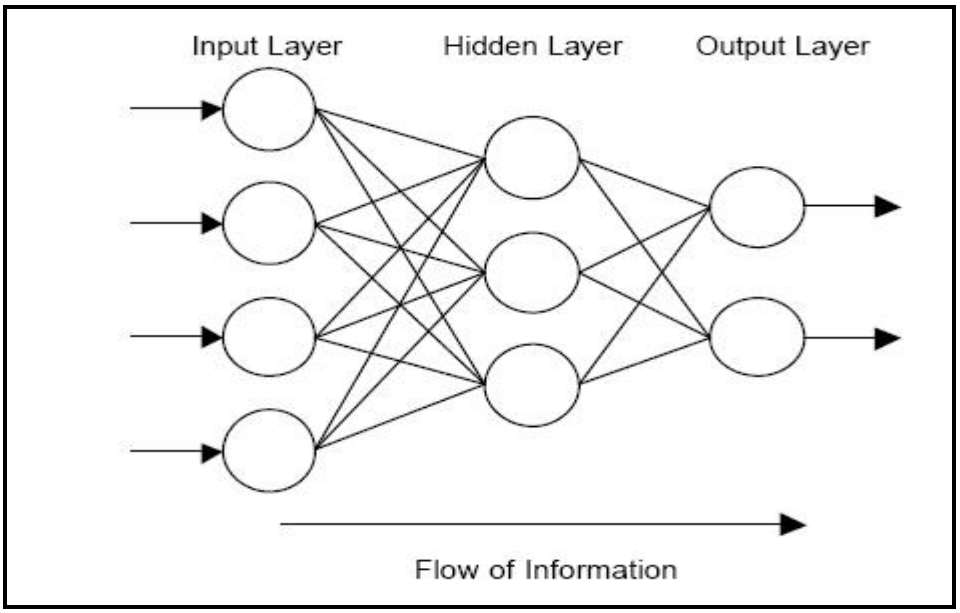

Figure 2. 2 The basic model of an ANN, (Fausett, 2001)

To capture the essence of biological neural systems, an artificial neuron is defined (Fausett, 2001) as follows:

- It receives a number of inputs (either from original data, or fromthe output of other neurons in the neural network). Each input comes via a connection that has a strength (or weight); these weights correspond to synaptic efficacy in a biological neuron.
- The weighted w_i sum of the inputs x_i is formed to compose the activation of the neuron. Gathering the weighs of the inputs according to:

$$Sum = \sum_{i=1}^{n} x_i w_i$$

- The activation signal is passed through an activation function *(y)* (also known as a transfer function) to produce the output of the neuron as follows:

$$y = f\left(\sum_{i=0}^{n} x_i w_i \right)$$

Figures 2.3 illustrates a summary of the principle work of an artificial neuron model, and a schematic illustrations of the basic features of a biological neuron and its artificial basic model is illustrated in Figure 2.4

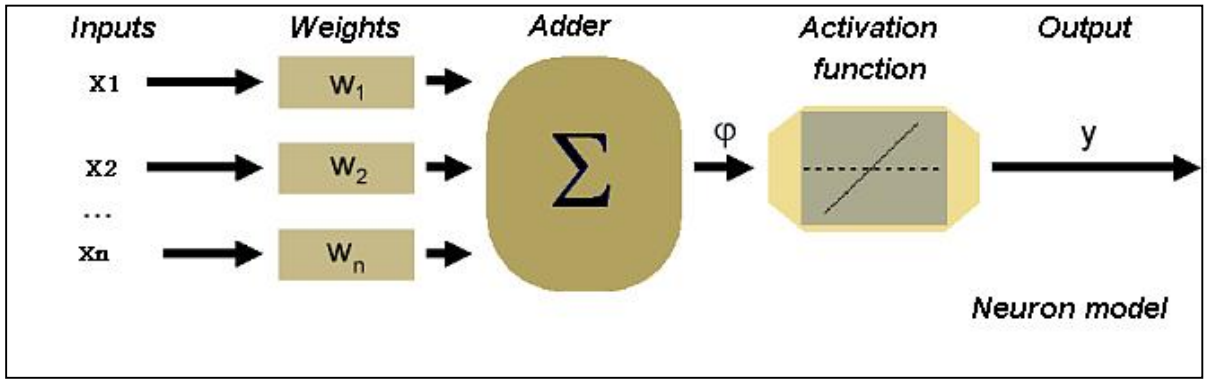

Figure 2. 3 summary of the principle work of Artificial Neuron Model, (Fausett, 2001)

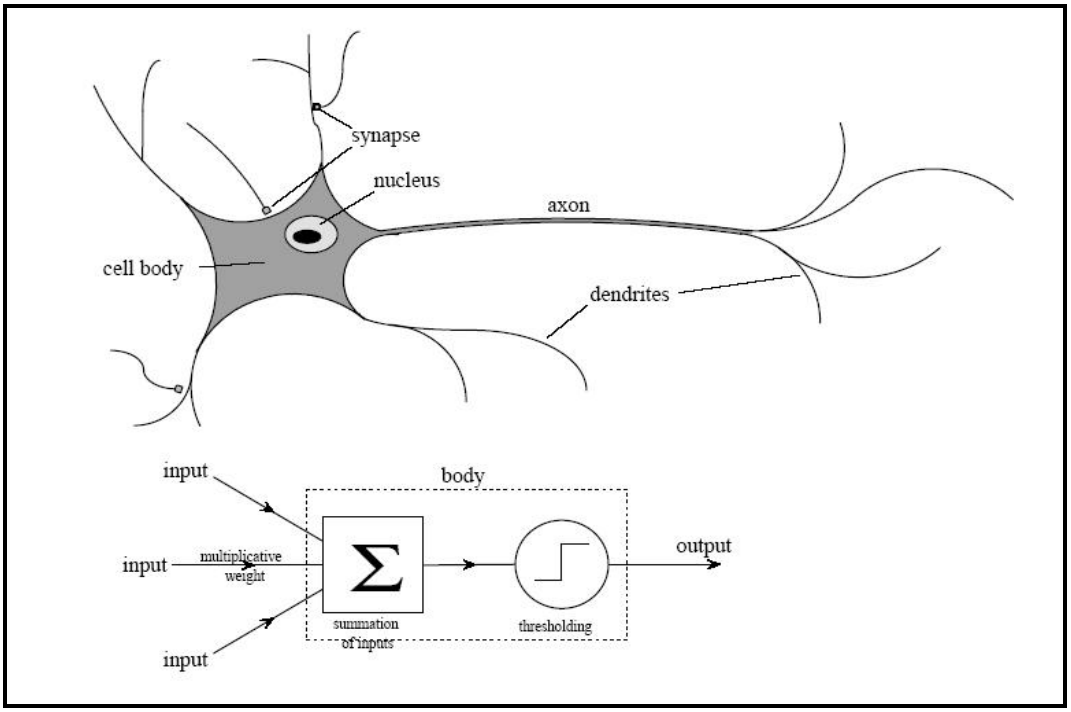

Figure 2. 4 Schematic Illustrations of the Basic Features of a Biological Neuron and its Artificial Basic Model, (Fausett, 2001)

2.1.3 Types of Activation Function

The output of a particular artificial neuron is determined by the type of activation function, as known sometimes transfer function (Freeman and Skapura, 1991).An activation function is used to limit the amplitude of the output of a neuron and is commonly known as a squashing function (Haykin, 2001). Neural Networks supports a wide range of activation functions. The most common types of activation functions are:

Linear activation function (LF): The outputs of a linear activation function, also known as a linear combiner, are determined as follows: $a = n$ where n is the sum of the weighted inputs. The graph of a LF function is represented as in Figure 2.5

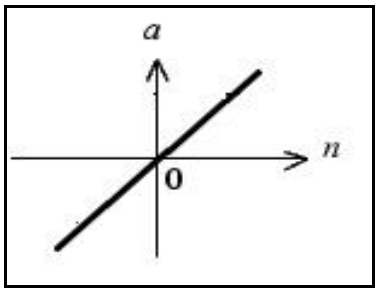

Figure 2. 5: Graph of a Linear Activation Function, (Hagan et al., 1996).

1. **Hyperbolic tangent sigmoid activation function (tanh)**: The hyperbolic tangent sigmoid activation function (tanh) has the following formula: $a = \frac{2}{1 + e^{-2n}} - 1$

where n is the sum of the weighted inputs. The resulted output from this function will be in a range between [-1,1] (Hagan et al., 1996). The graph of a tanh function is represented as in Figure 2.6

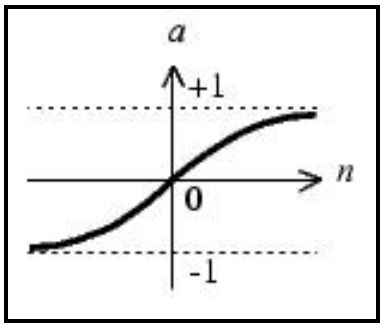

Figure 2. 6: Graph of a Hyperbolic Tangent Sigmoid Activation Function, (Hagan et al., 1996).

2.1.4 ANNs Architectures

ANNs architecture can be divided into two main categories based on their processing structure (known as topology): *feedback and feed-forward* neural networks (Mehrotra et al., 1996)**.**

In feedback neural networks, signals can travel in both directions. Thus the network has loops in its topology as shown in Figure 2.7(a). Networks with this structure can be very complicated and powerful, as they can continuously change state until an equilibrium is reached – this balanced state is then kept up until changes in the input data occur (Mehrotra et al., 1996).

In feed-forward neural networks signals are allowed to travel one way only; from input to output as shown in Figure 2.7 (b). There are no feedback (loop) connections. Feed-forward networks tend to be straight forward models that associate inputs with outputs (Mehrotra et al., 1996).

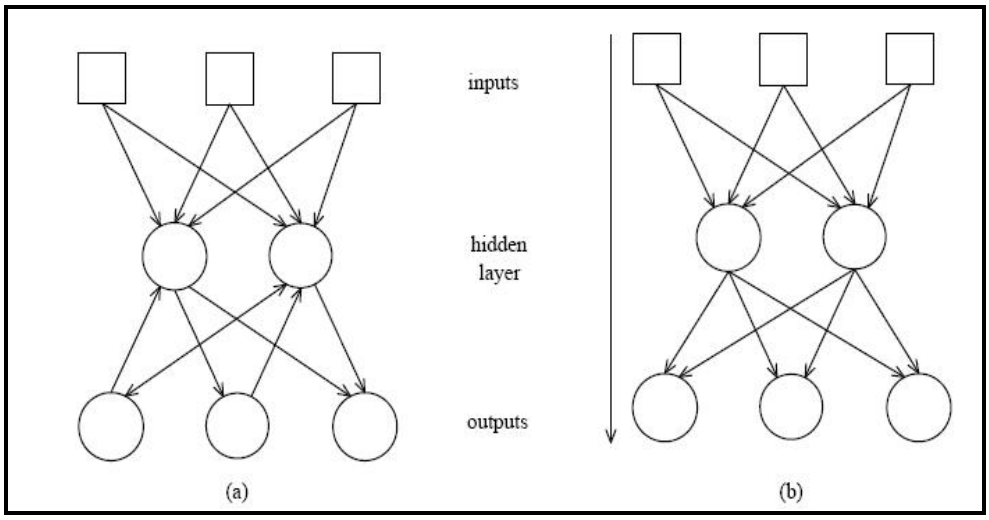

Figure 2. 7: A (a) Feedback and a (b) Feed-forward Neural Network, (Mehrotra et al., 1996).

These architectures can be further subdivided into two main classes based on the number of layers of processing nodes used in the model; single-layer perceptron and multi-layer perceptron (Fausett, 2001). In a single-layer perceptron there is a layer of input nodes and a layer of output nodes, which are the only processing nodes in the model. In a multi-layer perceptron (MLP) there are one or more hidden layers between the input and the output nodes.

2.1.4.1 The Single –Layer Perceptrons

In Single-Layer Perceptrons, or Perceptron, there is a layer of input nodes and a layer of output nodes, which are the only processing nodes in the model (Krose and Smagt, 1996). The general perceptron network architecture is shown in Figure 2.8.

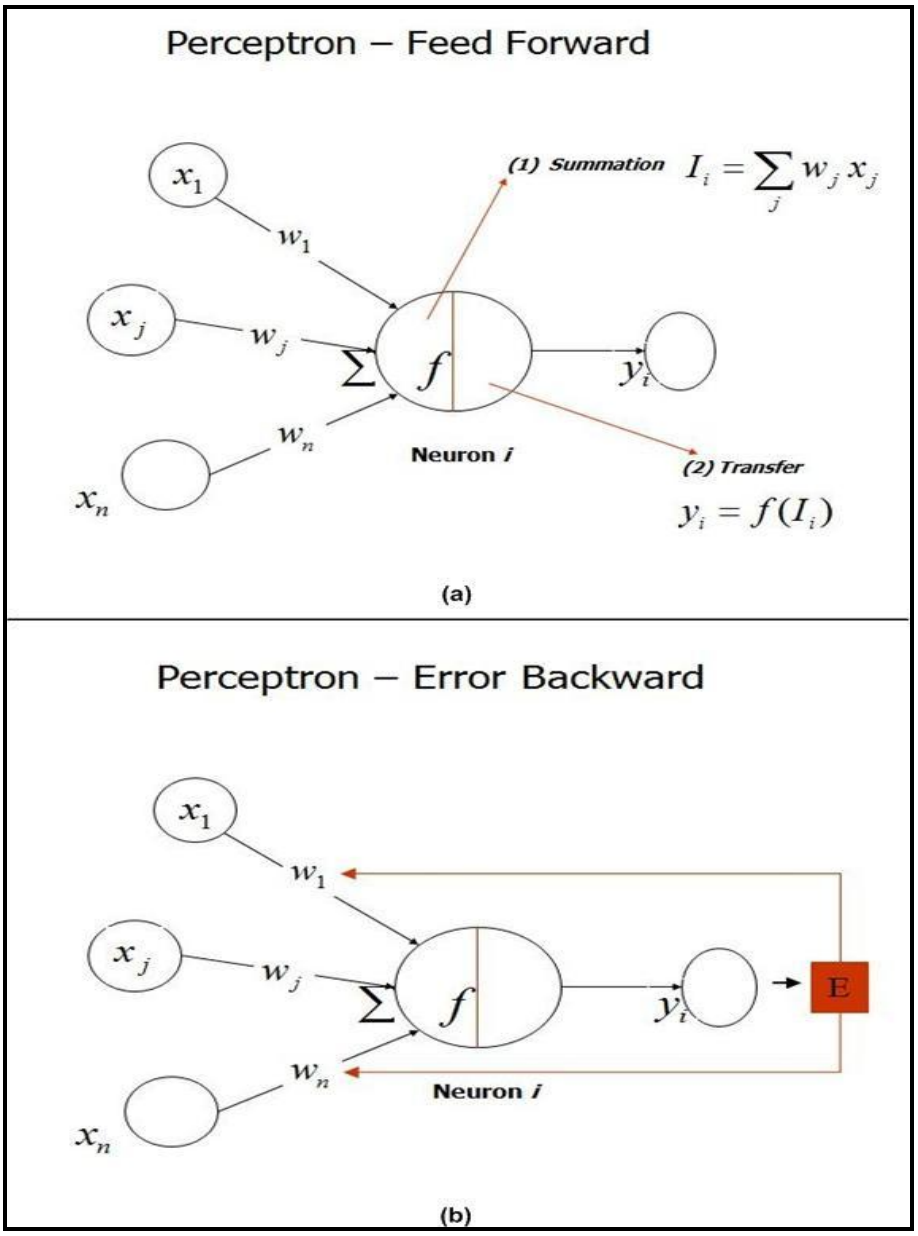

Figure 2. 8: Feed-Forward Network with Single-Layer Perceptron (a) Feed Forward Stage, (b) Error Backward Stage, (Krose and Smagt,1996).

This network is one of the simplest ANNs whose weights and biases could be trained to produce a correct target vector presented with the corresponding input vector (Veropoulos, 2001). Perceptrons are especially suited for simple problems in pattern

classification (linearly separable problems). This network is often trained with the perceptron learning rule. The perceptron learning rule is a learning rule for training single–layer hard–limit networks (Hagan et al., 1996).

A perceptron neuron uses the hard-limit (threshold) transfer function, the hard-limit transfer function is one of the transfer functions that maps inputs greater than or equal to 0 to 1, and all other values to 0 (Freeman and Skapura ,1991). In a perceptron neuron (which uses the hard-limit transfer function (hardlim)) each external input is weighted with an appropriate weight w_j, and the sum of the weighted inputs is sent to the hard-limit transfer function, which also has an input of 1 transmitted to it through the bias (Krose and Smagt, 1996).

The perceptron rule has two main stages; Feed-forward stage as in Figure 2.8(a), and error backward stage as in Figure 2.8(b), the details for each stage is as follow:

1. **Feed-forward stage**. Perceptron generates initially randomly weights, then the weights (w_j) and input values x_j are summed to compose activation function (v) as follows (Mehrotra et al., 1996):

$$I_i = w_0 + \sum_{j=1}^{m} w_j x_j$$

Where w_0 is called the bias that is a numerical value associated with the neuron which makes the neuron more powerful. It is convenient to think of the bias as the weight for an input x_0 whose value is always equal to one, so that:

$$I_i = \sum_{j=0}^{m} w_j x_j$$

The perceptron neuron produces a 1 if the net input into the transfer function is equal to or greater than 0; otherwise it produces a 0.

2. **Error backward stage.** In this stage the output error is calculate as the following:

$\varepsilon_i = t_i - y_i$, where t_i is the target output and y_i is the output, if the error is equal to zero then no weight update is performed, else the weight must be updated by applying the following:

$$w_j^{(new)} = w_j^{(old)} + \Delta w_j$$

$$\text{Where } \Delta w_j = \varepsilon_i \eta x_j$$

The term η is called the learning rate and is used to determine by how many the weights are changed at each step, ε_i is the error and x_j is the input correspond to that weight

2.1.4.2 Multi-Layer Perceptron (MLP)

In MLP, or multilayer feed-forward networks, there are one or more hidden layers between the input and the output nodes (Machado1996) as shown in Figure 2.2. The nodes belonging to these layers are usually referred to as hidden nodes. A simple network has a feed-forward structure: signals flow from inputs, forwarded through any hidden units, eventually reaching the output units. Such a structure has stable behavior. In other words, the feed-forward network is a layered network in which each layer only receives inputs from previous layers (Mehrotra et al., 1996). The feed-forward structures have proved most useful in solving non-linearly separable problems (Haykin, 2001).

The input layer in an MLP contains units that simply serve to introduce the values of the input variables. The hidden and output layer neurons are each connected to all of the

units in the preceding layer (Freeman and Skapura ,1991). When the network is executed (used), the input variable values are placed in the input units, and then the hidden and output layer units are progressively executed. Each of them calculates its activation value by taking the weighted sum of the outputs of the units in the preceding layer (Fausett, 2001). The activation value is passed through the activation function to produce the output of the neuron. When the entire network has been executed, the outputs of the output layer act as the output of the entire network.

2.1.4.3 Linearly and Non-Linearly Separable Problems

A problem is said to be linearly separable; if a straight line can be drawn to separate the input vectors into two categories, if more than one line is needed to separate the input vectors then we said that this problem is non-linearly separable (Freeman and Skapura ,1991). As was said before, the single layer perceptrons are powerful to be used with linearly separable problems, while MLP are used with non-linearly separable problems. A common example of a linearly separable problem is the AND Boolean function. Table 2.1 is the truth table of the AND Boolean function, represented as in Figure 2.9, we can notice that the AND operation is a linearly separable problem because the separable line classified three points (0,0), (0,1) and (1,0) as one group by, while the point (1,1) was classified as another group.

Table 2. 1: : Truth Table of the AND Boolean Function

X1	X2	X1 *AND* X2
0	0	0
0	1	0
1	0	0
1	1	1

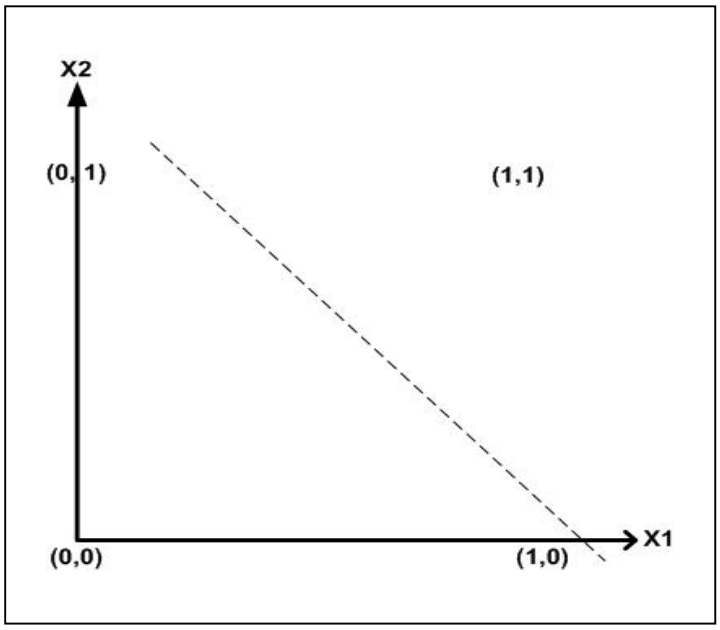

Figure 2. 9 :The AND Function is Linearly Separable , (Freeman and Skapura ,1991).

A common example of a non-linearly separable problem is the Boolean exclusive or (XOR) function, with the truth table shown in Table 2.2 , the desired computation is ‘’yes (1)’’ when one or the other of input is on, and ‘’no (0)’’ when they are both off or both on. So as illustrated in Figure 2.10, there is no clear line that can separate the two types of points.

Table 2. 2: Truth Table of the XOR Boolean Function

X1	X2	X1 *XOR* X2
0	0	0
0	1	1
1	0	1
1	1	0

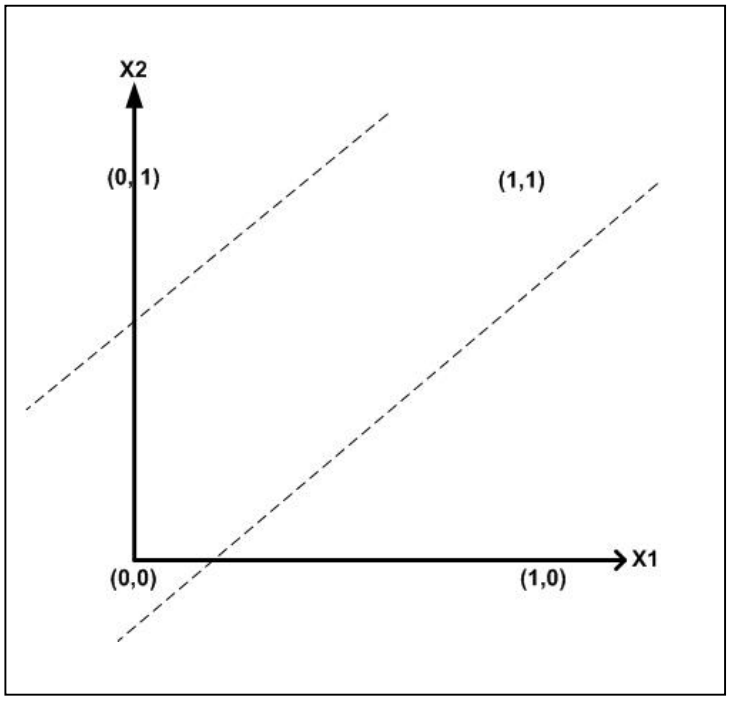

Figure 2. 10: The XOR Function is Non-Linearly Separable, (Freeman and Skapura ,1991).

2.1.5 The Back Propagation Algorithm

MLP neural networks have been applied successfully to solve difficult and diverse problem by training them in a supervised manner with highly popular algorithm known as Back Propagation which uses the data to adjust the network's weights and biases in manner that minimizes the error in its predictions on the training set (Fausett, 2001). If the network is properly trained, it has then learned to model the (unknown) function that relates the input variables to the output variables, and can subsequently be used to make predictions where the output is not known.

A Back Propagation network learns by example. You give the algorithm examples of what you want the network to do and it changes the network's weights so that, when training is finished, it will give you the required output for a particular input. In order to train the network you need to give it examples of what you want (the output you want (called the *Target*)) for a particular input as shown in Figure 2.11.

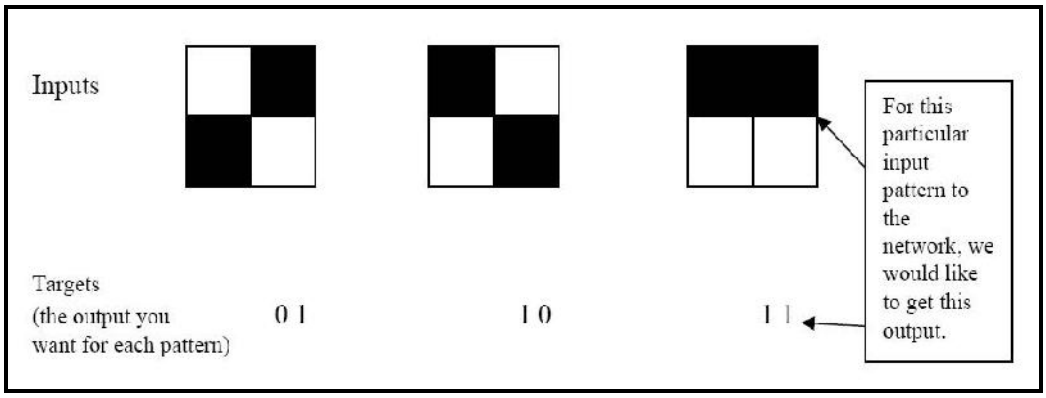

Figure 2. 11: a Back Propagation Training Set, (Krose and Smagt, 1996)

So, if we put in the first pattern to the network, we would like the output to be 0 1 as shown in Figure 2.12 (a black pixel is represented by 1 and a white by 0). The input and its

corresponding target are called a *Training Pair*. Once the network is trained, it will provide the desired output for any of the input patterns.

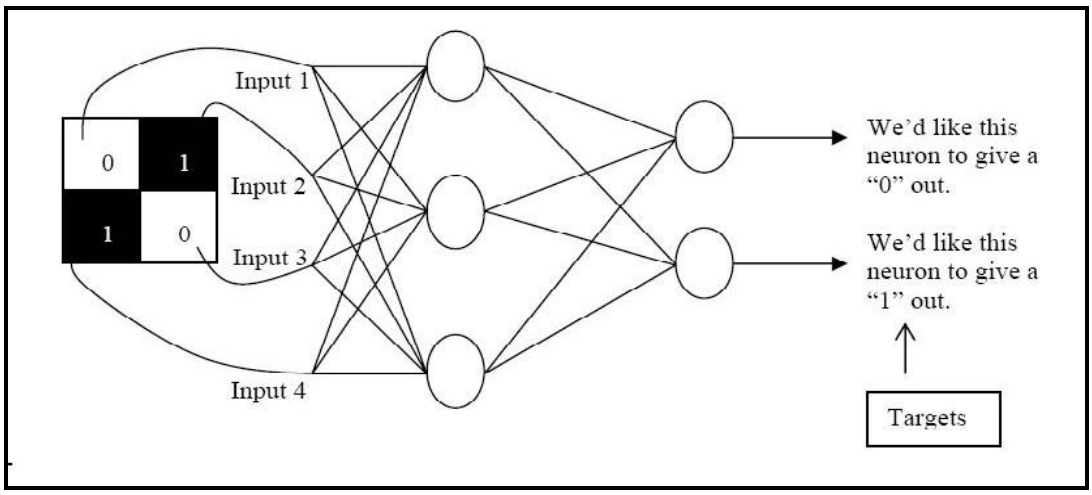

Figure 2. 12: Applying a Training Pair to a Network, (Krose and Smagt, 1996)

The training works as follows (Krose and Smagt, 1996):

- The network is first initialised by setting up all its weights to be small random numbers.
- Next, the input pattern is applied and the output calculated (this is called the *forward pass*). The calculation gives an output which is completely different to what you want (the Target), since all the weights are random.
- We then calculate the *Error* of each neuron, which is essentially: *Target – Actual Output* (i.e. What you want – What you actually get).
- This error is then used mathematically to change the weights in such a way that the error will get smaller. In other words, the Output of each neuron will get closer to its Target (this part is called the *reverse pass*).

- The process is repeated again and again until the error is minimal.

2.1.5.1 The Error Surface

Since Back Propagation iteratively searches for a set of weights that minimizes the error function **(E)** over all training pairs. Therefore MLP networks have error surface, which is in general complex and believed to have many local and global minima. This occurs because the algorithm always changes the weights in such a way as to cause the error to fall. But the error might briefly have to rise, as shown in Figure 2.13. If this is the case, the algorithm will get stuck (because it can't go uphill) and the error will not decrease further (Fausett, 2001). This requires rerunning the algorithm with new random initial weights and other network parameter hoping to reach the global minima in the new run. As a consequence, Back Propagation is never assured to find the global minima.

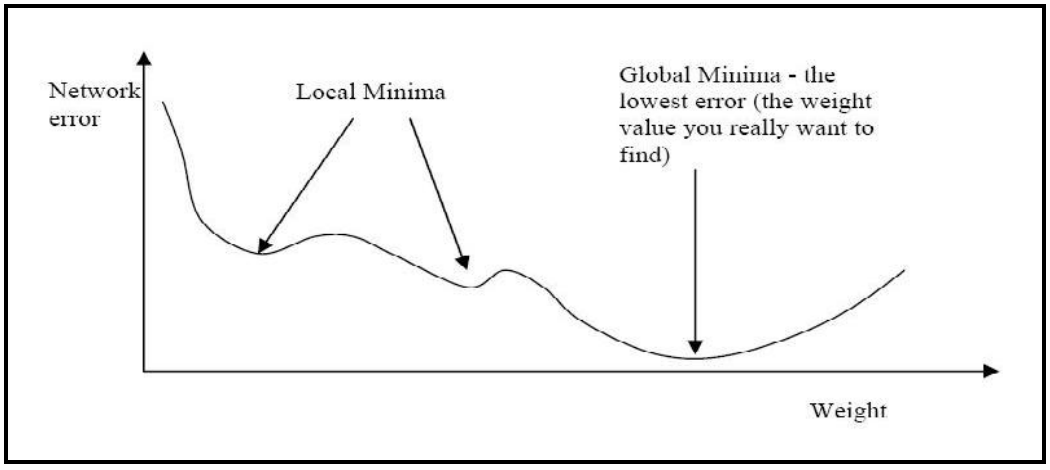

Figure 2. 13: An Error Surface, (Fausett, 2001).

2.1.5.2 The learning Rate

An important consideration in the Back Propagation algorithm is the learning rate ***η,*** which determines by how much we change the weights w at each step. If ***η*** is too small, the algorithm will take a long time to converge (reaching an optimal set of weights that give the solution) (Hagan et al.,1996). The Influence of choosing a small learning rate on the error surface is illustrated in Figure 2.14.

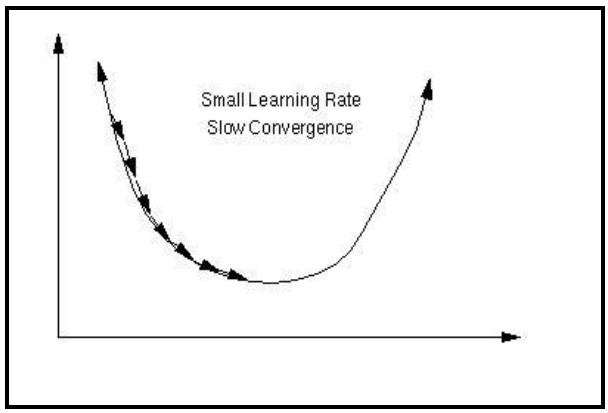

Figure 2. 14 : The Influence of Choosing a Small Learning Rate on the Error Surface, (Hagan et al.,1996)

Conversely, if ***η*** is too large, we may end up bouncing around the error surface out of control, the algorithm diverges. The influence of choosing a large learning rate is illustrated in **Figure 2.15**. The correct setting for the learning rate is application-dependent, and is typically chosen by experiment; it may also be time-varying, getting smaller as the algorithm progresses (Fausett, 2001).

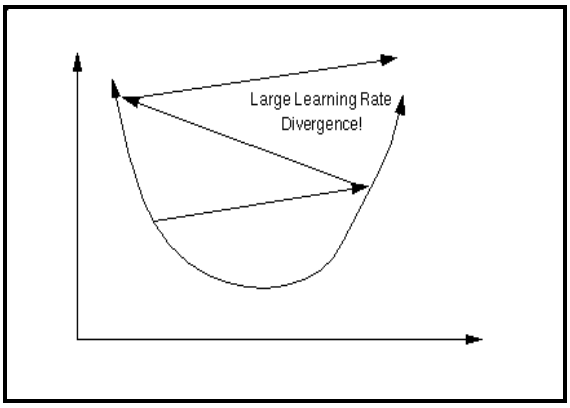

Figure 2. 15 : The Influence of Choosing a Large Learning Rate on the Error Surface, (Hagan et al.,1996)

2.1.5.3 The Back Propagation Steps in Details

The architecture of Backpropagation as shown in **Figure 2.16** work accordingly to the following :

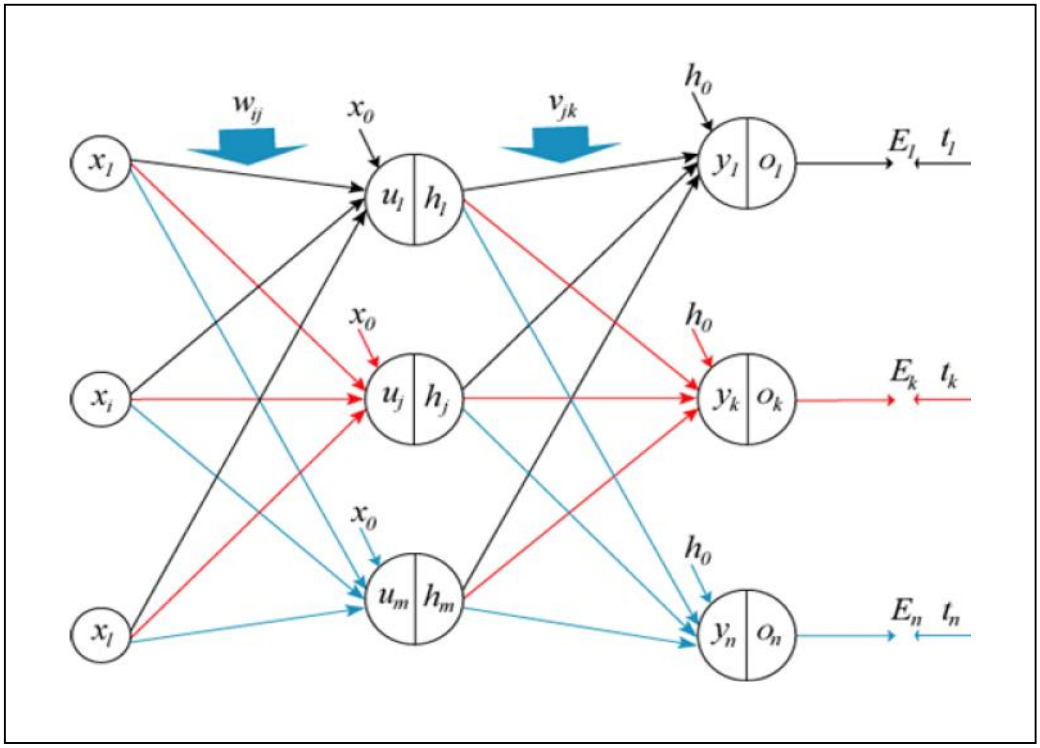

Figure 2. 16: The architecture of Backpropagation, (Krose and Smagt, 1996)

Step 1: Initialize the weights of the randomly with number between -1 and 1.

$$w_{ij} = random\ ([-1,+1]) \qquad 0 \le i \le l, 1 \le j \le m$$

$$v_{ij} = random\ ([-1,+1]) \qquad 0 \le j \le m, 1 \le k \le n$$

Step 2: Present the pattern *p* to the network, where p is one pattern from the whole pattern P:

$$p = (x_1, x_2, x_3, x_i, \ldots . x_l) \in R^l \qquad p \in P$$

Step 3: Compute the values of the hidden-layer nodes using the formula:

$$u_j = \sum_{i=0}^{l} w_{ij} \times x_i \qquad , h_j = f(u_j) \qquad , 1 \le j \le m$$

Step 4: Calculate the values of the output nodes using the formula:

$$y_k = \sum_{j=0}^{m} v_{ik} \times h_j \qquad , o_k = f(y_k) \qquad , 1 \le k \le n$$

Step 5: Compute the errors in the output layer using:

$$\delta o_k = (t_k - o_k) o_k (1 - o_k) \qquad , 1 \le k \le n$$

Step 6: Compute the errors in the hidden layer using:

$$\delta h_j = h_j (1 - h_j) \sum_{k=1}^{n} \delta o_k \times v_{jk} \qquad , 1 \le j \le m$$

Step 7: Adjust the weights between the output layer and the hidden according to:

$$v_{jk}^{(new)} = v_{jk}^{(old)} + \eta \times \delta o_k \times h_j \qquad ,0 \le j \le m, 1 \le k \le n$$

Step 8: Adjust the weights between the hidden layer and the input layer according to

$$w_{ij}^{(new)} = w_{ij}^{(old)} + \eta \times \delta h_j \times x_i \qquad ,0 \le i \le l \qquad ,1 \le j \le m$$

Step 9: Repeat step 2 through 8 for each element of training set, until the Total Mean Squared Error (TMSE) ≤ E or reach the maximum , E is an error threshold that is used to stop the training, error E defined as the difference between the desired t_k and current o_k output where (Hagan et al.,1996) :

$$TMSE = \frac{1}{n}\sum_{k=1}^{n}(t_k - o_k)^2$$

We notice from step 7 and 8 that these steps are the ones that change the weights of the network based on the following general formula:

$$w_{ij}^{(new)} = w_{ij}^{(old)} + \Delta w_{ij}$$

The term Δw_{ij}, is the amount of change in the current weight, this term in the back propagation algorithm is equal to; $\eta \times \delta o_i \times h_j$, where the term $\delta o_i \times h_j$ represent the partial derivative of the error with respect to the weight ($\frac{\partial E}{\partial w_{ij}}$) (Hagan et al., 1996).

The whole idea behind the partial derivative is; to gradually, but consistently, decrease the output error by adjusting the weights. The trick is to figure out how to adjust the weights. Intuitively, we know that if a change in a weight will increase (decrease) the error, then we want to decrease (increase) that weight. Mathematically, this means that we look at the derivative of the error with respect to the weight: $\frac{\partial E}{\partial w_{ij}}$, once we find this derivative, we will update the weight via the following:

$$\Delta w_{ij} = \eta \times \frac{\partial E}{\partial w_{ij}}$$

2.1.6 Resilient Back Propagation Algorithm (Rprop)

As discussed in subsection 2.1.5.1 a Back Propagation algorithm will experience a local minimum that occurs because the algorithm always changes the weights based on the value of the partial derivative. So in order to solve this problem; the Resilient Back Propagation (Rprop) was devised (Riedmiller and Braun, 1993), where it eliminates the harmful influence of the magnitude of the partial derivative. As a consequence, only the sign of the derivative is considered to indicate the amount of the weight update. To achieve this, an update-value Δ_{ij} is introduced for each weight w_{ij}, which solely determines the size of the weight-update (Riedmiller and Braun, 1993). This adaptive update-value evolves during the learning process**,** according to the following learning-rule:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^{+} \times \Delta_{ij}^{(t-1)} & , \; if \dfrac{\partial E}{\partial w_{ij}}^{(t-1)} \times \dfrac{\partial E}{\partial w_{ij}}^{(t)} \; \rangle \; 0 \\ \eta^{-} \times \Delta_{ij}^{(t-1)} & , \; if \; \dfrac{\partial E}{\partial w_{ij}}^{(t-1)} \times \dfrac{\partial E}{\partial w_{ij}}^{(t)} \; \langle \; 0 \\ \Delta_{ij}^{(t-1)} & , \; else \end{cases}$$

$$where \;\; 0 \langle \;\; \eta^{-} \;\; \langle \;\; 1 \langle \;\; \eta^{+}$$

The adaptation-rule works as follows: Every time the partial derivative of the corresponding weight w_{ij} changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value Δ_{ij} is decreased by the factor η^{-} (which was found by experiments that the best value for it is equal to **0.5**). If the derivative retains its sign, the update-value is slightly increased by the factor η^{+} (which was found by experiments that the best value for it is equal to **1.2**) in order to accelerate convergence in shallow regions.

Once the update-value for each weight is adapted, the weight-update itself follows a very simple rule: if the derivative is positive (increasing error), the weight is decreased by its update-value, if the derivative is negative, the update –value is added:

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & if \ \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)}, & if \ \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ 0, & else \end{cases}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$$

However, there is one exception: If the partial derivative changes sign, i.e. the previous step was too large and the minimum was missed, the previous weight-update is reverted:

$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)}, \quad if \frac{\partial E}{\partial w_{ij}}^{(t-1)} \times \frac{\partial E}{\partial w_{ij}}^{(t)} < 0$$

Due to that 'backtracking' weight-step, the derivative is supposed to change its sign once again in the following step. In order to avoid a double punishment of the update value, there should be no adaptation of the update-value in the succeeding step. In practice this can be done by setting $\frac{\partial E}{\partial wij}^{(t-1)} = 0$ in the Δ_{ij} adaptation rule above.

In the Rprop the weights and update-values are changed every time the whole pattern set has been presented once to the network (learning by epoch). Therefore, the term $\frac{\partial E}{\partial wij}^{(t)}$ denotes the summed gradient information over all patterns.

Rprop Algorithm

The following pseudo-code fragment shows the kernel of the **RPROP** adaptation and learning process. The **minimum (maximum)** operator is supposed to deliver minimum (maximum) of two numbers; the **sign** operator returns +1, if the argument is positive, -1, if the argument is negative, and 0 otherwise. While $\Delta_{\min} = 10^{-6}$ and $\Delta_{\max} = 50$.

For all weigths and biases {

if ($\frac{\partial E}{\partial w_{ij}}^{(t-1)} \times \frac{\partial E}{\partial w_{ij}}^{(t)} > 0$) **then** {

$\Delta_{ij}^{(t)} = \textbf{minimum} \quad (\Delta_{ij}^{(t-1)} \times \eta^{+} \quad , \Delta_{\max})$

$\Delta w_{ij}^{(t)} = -\textbf{sign} \ (\frac{\partial E}{\partial w_{ij}}^{(t)}) \times \Delta_{ij}^{(t)}$

$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$

}

else if ($\frac{\partial E}{\partial w_{ij}}^{(t-1)} \times \frac{\partial E}{\partial w_{ij}}^{(t)} < 0$) **then** {

$\Delta_{ij}^{(t)} = \textbf{maximum} \quad (\Delta_{ij}^{(t-1)} \times \eta^{-} \quad , \Delta_{\min})$

$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \Delta w_{ij}^{(t-1)}$

$\frac{\partial E}{\partial w_{ij}}^{(t)} = 0$

}

else if ($\frac{\partial E}{\partial w_{ij}}^{(t-1)} \times \frac{\partial E}{\partial w_{ij}}^{(t)} = 0$) **then** {

$\Delta w_{ij}^{(t)} = -\textbf{sign} \ (\frac{\partial E}{\partial w_{ij}}^{(t)}) \times \Delta_{ij}^{(t)}$

$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$

}

}

2.1.7 Designing Issues of Artificial Neural Networks

There are several issues to be considered when designing a neural network application, aiming to achieve a good generalization performance – i.e. to have the outputs of the network approximate well target values given inputs that are not in the training set. These issues will be discussed in the following subsections.

2.1.7.1 Selection of an ANN Topology

When it comes to decide what ANN topology to be used and which training algorithm to apply to this structure, there is no easy answer. Selection of a suitable combination of these two design components is often done experimentally and it is problem dependent (Mehrotra et al., 1996).

Other issues closely related to the network topology include the number of hidden layers and hidden units to be used. In a wide variety of applications a linear model can be sufficient for developing a good classification system. In that case no hidden layers will be needed (Fausett, 2001). However, in applications where a nonlinear model is needed, a MLP with one hidden layer with arbitrary large number of units typically suffices for a good approximation. The optimum number of hidden units depends on the number of input features, the size of the training set, the amount of noise presented in the data, the architecture and the choice of functions and parameters in the algorithm (Veropoulos, 2001). Although a lot of attempts have been made in order to determine a general rules that give the most appropriate number of hidden units to be used for training the network (Mehrotra et al., 1996), none of them can be considered reliable enough or true for any situation.

Another issue is to determine how many training samples are required for successful learning, and how large a neural network is required for a specific task, these are solved in practice by trial and error (Mehrotra et al., 1996). These issues are complex because there is considerable dependence on the specific problem being attacked using a neural network (Veropoulos, 2001).

2.1.7.2 Pre-Processing and Post-Processing

The input data should be in such a form that a neural network can process it, this means numeric values. Furthermore, the numeric input values may be pre-processed in a way that the network's learning task is easier. In many practical applications, the choice of the pre-processing method will be one of the most significant factors in determining the performance of the ANN (Hagan et al., 1996). Pre-processing is always performed before the training process begins. Sometimes the output data are also processed. This is called post-processing which is converting back the output into the same units that were used in the original dataset.

One commonly used pre-processing method is data standardization; standardizing a vector most often means subtracting a measure of location and dividing by a measure of scale. For example, you might subtract the mean and divide by the standard deviation, thereby obtaining a "standard normal" random variable with mean 0 and standard deviation The average (mean) is a measure of the middle value of the data set. If n numbers are given, each number denoted by $\boldsymbol{a_i}$, where i = 1,..,n, the mean is the [sum] of the $\boldsymbol{a_i}$'s divided by n or:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{N} a_i$$

While the standard deviation shows how much variation or "dispersion" there is from the "average" (mean). A low standard deviation indicates that the data points tend to be very close to the mean, whereas high standard deviation indicates that the data are spread out over a large range of values (Ghahramani, (2000). The standard deviation S_N is defined as follows:

$$S_N = \sqrt{\frac{1}{n} \sum_{i=1}^{N} \left(x_i - \overline{x} \right)^2}$$

In order to obtain a dataset with zero mean and unit standard deviation, we use the following :

X_i = Value of the raw input variable X for the i^{th} training case.

S_i = Standardized value corresponding to X_i.

N = Number of training cases.

Standardize X to mean 0 and standard deviation 1 by:

S_i= (Xi – mean(X)) / Standard Deviation(X).

Note that statistics such as the mean and standard deviation are computed from the training data, not from the validation or test data. The validation and test data must be standardized using the statistics computed from the training data.

2.1.7.3 Generalization Performance of an ANN

One of the major advantages of neural networks is their ability to generalize; this means that a trained network could classify data that it has never seen before from the same class as the learning data. In real world applications developers normally have only a small part of all possible patterns for the generation of a neural network (Hagan et al.,1996). To reach the best generalization, the dataset should be split into three parts:

- **The training set** that is used to train a neural network. The error of this dataset is minimized during training.
- **The validation** set that is used to determine the performance of a neural network on patterns that are not trained during learning.
- **A test set** for finally checking the over all performance of a neural network.

The learning should be stopped in the minimum of the validation set error. At this point the network generalizes best. When learning is not stopped, overtraining occurs and the performance of the network on the whole data decreases, despite the fact that the error on the training data still gets smaller. After finishing the learning phase, the network should be finally checked with the third data set, the test set (Veropoulos,2001). An example of an overfitted network is illustrated in Figure 2.17

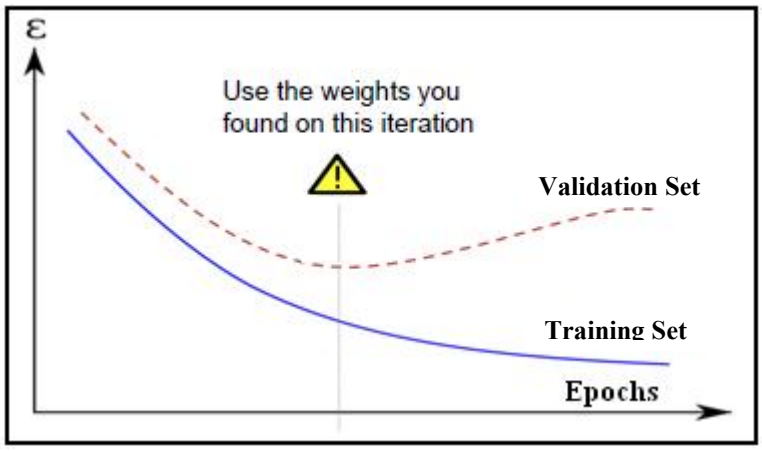

Figure 2. 17: Overfitting in Neural Networks, Training error is shown down and validation error is shown up, (Veropoulos, 2001).

Early Stopping

This technique is used for improving the generalization of a neural network during training and preventing it from overfitting (Haykin 2001). It achieves that by splitting the available data into three subsets which are used separately for training, validation and testing. These sets are used in the following manner; in each iteration through the training set, the ANN runs a test over the validation set; this is done iteratively while the performance of the network continues to improve. When the performance does not show further improvement with more iteration then the training is completed and the ANN runs a test over the test set. The result gives the final performance of the classifier (Veropoulos,2001).

The basic concept is that the performance of the ANN against the training set will continue to improve with more training, and this is also true for its performance against the validation set (Machado1996). However, if the ANN starts overfitting the training set, its performance then against the validation set will start to decrease. Therefore, the validation error will increase; this is an indicative of a poor generalization and can be used for

stopping the classifier from further training (Haykin 2001). The weights giving best performance against the validation set can then be accepted as the set of weights that gives the ANN best generalization ability. Since the error surface is not always smooth, it is not certain at this point of the training if the validation error is the global minimum (Hagan et al.,1996). So, it is common practice to continue training for several iterations (based on the judgment of the designer) in case the ANN performance against the validation set improves again. At the same time the configuration and set of parameters of the ANN that gave best results until this point are always stored (Veropoulos,2001).

Because the validation set has been utilized during the training process, after the training is finalized, it is important to test the trained ANN against a separate set (the test set), which hasn't been used at all during training and will give an unbiased estimate of the generalization error of the network(Hagan , et al.,1996)..

2.2 Using Neural Networks in Medical Problems.

As we have seen in the previous chapter, neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns that are too complex to be noticed by either humans or other computer techniques, therefore due to their capabilities they have been used widely in the medical field, where one of the most important problems in the medical domain, in general, is the subjectivity of the specialist. It can be noted, in particular in pattern recognition activities that the experience of the professional is closely related to the final diagnosis or the treatment a physicians may give to a patient, and this is due to the fact that the result does not depend

on a systematized solution but on the interpretation of the patient's record. Therefore a great deal of research has been directed toward applying ANN in the medical fields. Section 2.2.1 discusses some of the applications of neural networks in the medical domain in general, where section 2.2.1 discusses in particular, some of the applications of neural networks in the diabetes domain.

2.2.1 Applications of Neural Networks in the Medical Field

Previous studies sought ways of capitalizing the use of neural networks in several disease, it proved that the Information Technology helpful in breast cancer where it can improve the results of breast cancer screenings by evaluating the performance ANN to predict an outcome (cancer/not cancer). Paulin and Santhakumaran (2011) presented a study on the classification of breast cancer using feed forward ANNs; the performance of the network is evaluated using Wisconsin Breast Cancer Data set **(WBCD)** where an accuracy of 99.28% using the Levenberg-Marquardt **(LM)** training algorithm was obtained. Also (Kıyan and Yıldırım, 2003) evaluated the performance of the Radial Basis Network **(RBF)**, General Regression Neural Network **(GRNN)**, MLP and Probabilistic Neural Network **(PNN)** on the WBCD and they concluded that the most suitable neural network model for classifying WBCD is GRNN with accuracy 98.8% while the classification performances for the other type of neural network were 96.18% for RBF, 97.0% for PNN, and 95.74% for MLPs.

In hepatitis domain, several studies proved that neural networks are useful for the differentiation of two forms of chronic active hepatitis, Nakano et al, (1996) investigated

the ability of neural networks to discriminate mild and severe chronic active hepatitis where the result showed a network capable of correctly predicting the diagnosis for 78% of the cross-validation group.

Bascil and Temurtas, (2009) presented a study for hepatitis diagnosis using a multilayer neural network as a machine learning technique and LM algorithm as training algorithm, the architecture was on the data set that was taken from University of California Irvine **(UCI)** machine learning database, the study obtained a classification accuracy of 91.87% via ten-fold cross validation.

On other hand there are some studies showed the ability of neural networks in predicting diseases complication, Vahdani et al. (2009) considered in their study on predicting the presence or absence of cirrhosis in patients with chronic hepatitis B; they concluded that in patients with chronic hepatitis B, if the ANN value of certain laboratory manifestations is negative, there will be a 95% chance of having a negative liver biopsy.

In heart disease there are several diseases that share the same symptoms which increase the ability of wrong diagnosis so that ML can help in making correct diagnoses. Yan et al. (2006) presented a multilayer perceptron based decision support system to support the diagnosis of heart diseases; the system was applied to five different heart diseases. The results showed that the proposed system could achieve very high diagnosis accuracy 90% proving its usefulness in support of clinical diagnosis decision of heart diseases

AlTimemy and Al Naima (2010) in their study presented the prediction of kidney dysfunction using different neural network approaches, PNN, SOM and MLP neural network trained with back propagation algorithm; these networks are used as a classifiers to predict whether kidney is normal or it will have a dysfunction applied on a data set

collected from one of the private clinical laboratories in Baghdad, In this study the PNN gave more accurate prediction of kidney dysfunction with accuracy 99% while the accuracy of SOM was 98% and the accuracy for MLP was 95%.

2.2.2. Applications of neural networks in the diabetes Field:

Many researchers give attention to diabetes since it is a chronic disease, for that it is include a multiple fields that possible to be computerized.

(Kayaer and Yildirim ,2003) applied Three different neural network structures, which are multilayer perceptron (MLP), radial basis function (RBF) and general regression neural network (GRNN) to the Pima Indians Diabetes (PID) medical data obtained from UCI machine learning repository. They achieved 80.21% classification accuracy using general regression neural network (GRNN) and they have also reported 77.08% classification accuracy using multilayer neural network with LM algorithm.

(Polat and Gunes,2007) have reported 89.47% classification accuracy using 10 fold cross validation on Pima Indian diabetes disease diagnosis using principal component analysis (PCA) and adaptive neuro-fuzzy inference system (ANFIS).

(Temurtas et al, 2009) in their study realized a comparative pima diabetes disease diagnosis For this purpose, a multilayer neural network structure which was trained by Levenberg–Marquardt (LM) algorithm and a probabilistic neural network structure were used, the study achieved 79.62 % accuracy using 10 fold cross validation approach and it achieved 82, 37% accuracy using the conventional validation approach (one training and one test).

On another hand (Abu Zitar and Al-Jabali, 2005) proposed a model whose aim is to predict correctly the next values of glucose levels (NGL) passed on previous observations for the patients state using Levenberg- Marquardt (LM) training algorithm of multilayer feed forward neural network (NN).

(Zainuddin et al ,2009) in their investigation use a separate blood glucose measurement in the morning, afternoon, evening and night intervals from one patient covering a period of 77 days , based on the PCA and a predictor based on wavelet neural network are used. Theirs experimental results showed that the proposed expert system is a powerful model for the blood glucose prediction.

Another researches give attention in the psychological state of diabetic patients (Manda et al ,2005) purpose feedforward back propagated neural network with graphical user interface based icons. They collected data from 241 diabetic patients with 33 variables and found that four variables are very important for predicting the well being (age, gender, weight, fasting plasma glucose). Predefined values were considered as set of inputs and found the output for measuring the well being which are Energy, Positive Well being, Depression and Anxiety

Since there are multiple type of insulin treatment as shows bellow:

1. Short-acting insulin mixed with intermediate-acting insulin, given twice daily before meals
2. Short-acting insulin, given three times daily AND intermediate-acting insulin, given at bedtime
3. intermediate-acting insulin, given once daily

(Gogou et al ,2001) proposed a system will further facilitate decision making for diabetic patient management by insulin administration based on a neural network (NN) approach. The factors participating in the decision making were among others diabetes type, patient age, current treatment, glucose profile, physical activity, food intake, and desirable blood glucose control. The resulting system was trained with 100 cases and tested on 100 patient cases. The system proved to be applicable to this particular problem, classifying correctly 92% of the testing case

Chapter Three

3. Data Preparation and System Architecture

ANN performs quit well with complex tasks such as classification problems using abstract data. ANN has been applied to many areas of the medical field for analysis of various diseases and conditions. Therapeutic Decision Support System for Type 2 Diabetes using ANN is a fairly new area, and not much analysis has been performed with ANN. In this chapter section 3.1 will discuss how data was collected and prepared in order to build the diabetes management CDSS. Section 3.2 will discuss the system architecture for the diabetes management CDSS

3.1 Data Preparation

When a patient reports to a physician, a large number of possibly relevant inputs must be considered during the session. A physician reaches an accurate diagnosis or treatment decision based upon observations, the patient's answers to questions, and physical examinations or lab results.

In this thesis, the diabetes datasets used for testing and training the system consists of a total of 228 cases, in which each medication type has a total of 76 cases, gathered from Jordan University hospital, located in Amman, Jordan, we have trained and tested the system using these medical records. Moreover 25 variables essential to give the stabile treatment these variables can be divided into four categories:

i. Basic information of a patient (6 factors in total).

ii. The patient's previous status and test :(4 factors in total).

iii. Diabetes Complication (7 factors in total)

iv. Lab results and physical examination (8 factors in total).

In figure (3.1) the datasheet that was used in gathering the patent's variables is illustrated; We can notice from the figure that there are some attributes were assigned to have a yes or no value to indicate the presence or absence of an attribute while another attribute assigned to have one value such as gender and previous treatment the value of the gender have to be either female or male and the value of the previous treatment have to be single medication ,dual medication or insulin therapy. The remainder attributes Age and all Basic information, diastolic blood pressure, systolic blood pressure, and all blood lab results were assigned the actual values the patient has for these attributes.

Patients Data Set

System Output: Treatment Type:

I. Single medication
II. Dual medication
III. Insulin /insulin + Oral medication

System Input:

i. Patient's Basic Information:

- *Age:* _____
- *Gender: Male / Female*
- *Weight:* _____
- *BMI:* _____
- *Height:* _____
- *Duration:* _____

ii. Diabetes Complication

- *Hypertension: Yes/No*
- *Dyslipidemia: Yes/No*
- *Diabetic Neuropathy : Yes/No*
- *Hypoglycemic Attack : Yes/No*
- *Diabetic Nephropathy : Yes/No*
- *Diabetic Retinopathy : Yes/No*
- *Cardiac Heart Disease : Yes/No*

iii. The patient's previous status and test :

- *Previous weight:* _____
- *Previous Treatment Type:* _____
- *Previous HBA1C:* _____
- *Previous FBS:* _____

iv. Lab results and physical examination:

- *Current HBA1C:* _____
- *Current FBS:* _____
- *Blood Pressure:* ____ /____
- *Cholesterol:* _____
- Triglyceride: _____
- Low Density lipoprotein: _____
- High Density lipoprotein: _____

Figure 3. 1 Patient's Datasheet

3.2 System Architecture

The following subsections explain how the input variables that were used in the training and testing process were encoded for building the diabetes treatment CDSS, and how the number of hidden layers and hidden neurons in each layer were chosen.

3.2.1 Input Variables Classes Encoding Scheme

All neural networks take numeric input, therefore the 25 variables were encoded into numeric values, as well as the predicted treatment type using the following encoding schemes:

i. Numerical variables such as Age and all Basic information, diastolic blood pressure, systolic blood pressure, and all blood lab results were reserved the same with no changes since they are already in a numerical format.

ii. Variables with two independent attributes, such as the gender For instance, 1 represents female and 0 male, some attributes from the diabetes complication are encoded with binary values (0, 1). 1 is adopted for the mentioned categories for the presence of the attribute in that category, and 0 for the absence of it.

iii. Variables with three independent attributes, such as previous treatment are encoded using the ternary values (1,2,3); with 1 representing that the patient use single medication as treatment, 2 representing that the patient using dual medication, and 3 for the patient treated by insulin.

After encoding all variables, the training dataset was standardized to have a zero mean and a unit standard deviation as explained in subsection (2.1.7.2), and based on the information that was collected from the training dataset during the standardization; the validation and test datasets were standardized also to have a zero mean and a unit standard

deviation. As we mentioned before, the standardization method must be applied for both input and output variables.

3.2.2 Number of Hidden Layers and Hidden neuron

Optimizing the number of hidden layer and neurons in each hidden layer for a feed forward neural network remains one of the unsolved tasks in this research area. Thus several researchers have proposed some general rules for determining an optimal number of hidden neurons for any application. One of the useful rules is; One hidden layer in general is enough and for the first hidden layer add the number of neurons in the input layer with the number of neurons in the output layer and divide the result by two, another known rule is; if more than one hidden layer must be used to achieve a less generalization error then the number of neurons in each hidden layer except the first one will be the number of neurons in the previous layer divided by two (Xu and Chen, 2008). Moreover Kavzoglu as cited in (Yan et al., 2006) has shown that in most situations, there is no way to determine the best number of hidden layers without training several networks and estimating the generalization error of each, because the best number of hidden layers and units depends mainly in a complex way on; the numbers of input and output units, the number of training cases and the complexity of the classification to be learned (Xu and Chen, 2008).

Based on this, we used an iterative process to determine the best number of hidden layers and neurons in each hidden layer. In the iterative process a ten- fold cross validation technique (more details will be provided in chapter 4) is used to choose the best neural model that will provide the highest classification accuracy, the whole process works as follow:

Step1. Starting from one hidden layer; the network architecture has been tested with one hidden neuron up to 25 neurons.

Step2. Add another layer; the number of neurons in this layer will be half the number of neurons in the previous layer.

Step3. Repeat step two until the number of the neurons in the layer is one.

Chapter Four

4. Experimental Results and Evaluation

In this chapter, Section 4.1 introduces the evaluation method that was used to assess the generalization of the ANN models. Section 4.2 presents multiple ANN architectures that were built in order to choose the best ANN model for building the CDSS for diabetes treatment. Finally Section 4.3 introduces the detailed results for the best ANN model we obtained in Section 4.2.

4.1 Cross validation

Cross validation is a technique used for estimating the performance of a predictive model, also it can be referred as estimate of accuracy, sometimes referred as rotation estimation, and it is determined by the overall number of correct classifications divided by the total number of instances in the dataset (Yan et al., 2006). One round of cross-validation involves partitioning a sample of data into subsets, performing the analysis on one subset (called the training set), and testing the analysis on the other subset (called the testing set). As we discussed before in subsection 2.1.7.3, ANN go through overfitting, therefore a third subset called the validation set is used to avoid overfitting. Moreover to reduce variability and to avoid bad splits that may produce overfitting , multiple rounds of cross-validation are performed using different partitions, and the tested results are averaged over the rounds.

In this thesis a ten-fold cross validation was used to access the generalization of the network models that were built. A percentage of 80%, 10% and 10% data split of the complete data samples was applied to represent the training, validation and test subsets

respectively. This provides for each fold a total of 186, 21and 21 sample cases for the train, validation and test subsets respectively. Each medication type in the training set will be represented by 62 samples of the total samples of the training set. Also each medication type in the validation set will be represented with 7 samples and 7 samples for the test set.

Classification Accuracy

The classification accuracy is the percentage of the cases that correctly classified, in ANN classification accuracy is the most important issue since this performance metric give some indication of the ANN performance. Accuracy (***A)*** of an individual testing dataset depends on the number of samples correctly classified and is evaluated by the formula:

$$A = (T/N)* 100\%$$

Where

T: is the number of cases correctly classified.

N: is the total number of cases.

4.2 Models' Building

In order to investigate the best system architecture for building the CDSS for diabetes treatment; multiple system architectures were built using the iterative process that was explained in subsection 3.2.2. The different architectures were compared based on the classification accuracy which each architecture has obtained. The model with the highest classification accuracy (less generalization error) was chosen as the final model for building the CDSS.

The different ANN architectures have been built and tested using MATLAB version 2010a, which was run under windows Microsoft Windows XP Professional version 2002 OS with an Genuine intel ® CPU , and 2.49 GB of RAM. .

In this thesis for all models the following parameter are used:

- A Feedforward Backpropagation neural network is used for building all models.
- The number of neurons in the input layer is 25 (representing number of features for each sample).
- The number of neurons in the output layer is 1 (representing one class of the treatment the neural will generate).
- The training algorithm that was used for training the models is Resilient Backpropagation.
- The activation function that is used for all the hidden layers is HTSF and the activation function that is used for the output layer is LF.

• The following values were used 0.08, 0.0000001, and 6 for the delta initial, the performance goal error and the number of validation checks to avoid the overfitting of the network respectively.

4.2.1 Models' Building using One Hidden Layer

Since the input layer contains 25 neurons, we built network architecture with one hidden layer and with one up to 25 neurons.

Table (4.1) illustrates the results of 25 different architectures, the best architecture that produced the highest classification accuracy is (25-11-1) ; this architecture represents 25 input variables, one hidden layer of 11 neurons and one output layer, the classification

accuracy for this architecture is 88.5714%. All the remaining architectures produced low classification accuracy varying in an interval [73.33-87.14].

Table 4.1 Classification Accuracy with one hidden layer

Network Architecture	Classification Accuracy
25-1-1	80.4762
25-2-1	82.8571
25-3-1	73.3333
25-4-1	86.6667
25-5-1	86.6667
25-6-1	84.2857
25-7-1	81.9048
25-8-1	81.4286
25-9-1	86.1905
25-10-1	84.7619
25-11-1	88.5714
25-12-1	85.7143
25-13-1	85.2381
25-14-1	85.2381
25-15-1	86.6667
25-16-1	79.5238
25-17-1	84.2857
25-18-1	78.5714
25-19-1	83.8095
25-20-1	86.6667
25-21-1	87.1429
25-22-1	76.6667
25-23-1	77.1429
25-24-1	85.7143
25-25-1	80.4762

4.2.2 Models' Building using Two Hidden Layers

For the first hidden layer we obtained 25 architecture, so by applying step 2 from the iterative process, number of neurons in the second layer will be as follows:

i. When the number of neurons in the first hidden layer was even, the number of neurons in the second hidden layer will be the number of hidden neuron in the first layer divided by 2

ii. When the number of neurons in the first hidden layer was odd, the number of neurons in the second layer once will be the ceiling of the number of hidden neuron in the first layer divided by 2, another time it will be the floor of the number of hidden neuron in the first layer divided by 2.

Consequently, 37 different architectures will be tested when the number of the hidden layers is two. As noticed from table (4.2), the best classification accuracy is 86.6667% it is obtained when the network architecture is 25-9-5-1 and when the network architecture is 25-13-7-1. The results for the other architectures vary in an interval [69.04-85.2381].

Table 4.2: Classification Accuracy with two hidden layer

Network Architecture	Classification Accuracy
25-1-1-1	77.619
25-2-1-1	79.5238
25-3-1-1	82.8571
25-3-2-1	73.8095
25-4-2-1	76.6667
25-5-2-1	83.8095
25-5-3-1	79.5238
25-6-3-1	83.3333
25-7-3-1	78.5714
25-7-4-1	82.8571
25-8-4-1	80.9524
25-9-4-1	81.4286
25-9-5-1	86.6667
25-10-5-1	80.9524
25-11-5-1	85.2381
25-11-6-1	74.2857
25-12-6-1	79.5238
25-13-6-1	78.0952
25-13-7-1	86.6667
25-14-7-1	74.7619
25-15-7-1	80.4762
25-15-8-1	78.0952
25-16-8-1	82.381
25-17-8-1	81.9048
25-17-9-1	78.0952
25-18-9-1	77.1429
25-19-9-1	70.9524
25-19-10-1	79.5238
25-20-10-1	82.381
25-21-10-1	81.9048
25-21-11-1	78.0952
25-22-11-1	84.2857
25-23-11-1	82.381
25-23-12-1	83.8095
25-24-12-1	83.3333
25-25-12-1	80
25-25-13-1	69.0476

4.2.3 Models' Building using three Hidden Layers

We obtained 37 different architectures when the number of hidden layers was two, so that when a third hidden layer is added, the number of possible architecture will be increase and it will be hard to manage , to decrease the number of possible architecture we need to modify the step 1 in iterative process (mentioned before) to be as follow :

Step1. We start testing the network architecture with only one hidden layer; the following equation will be applied to find the number of neurons in the first hidden layer:

$$n_f = (n_i) / 2$$

Where n_f is the number of neurons in the first hidden layer, n_i is the number of neurons in the input layer.

Step2. Add another layer; the number of neurons in this layer will be half the number of neurons in the previous layer.

Step3. Repeat step two until the number of the hidden neurons in the layer is one.

Since the input layer contains 25 neurons, the number of neurons in the first hidden layer will be 12.5 after applying the equation in step 1 from the new iterative process. Because the result is not a fix number, applying the ceil operation will produce 13, applying the floor operation will produce 12, therefore the network can be trained with two different numbers of neurons, and to make the testing more precise another number is used, which is 11. Accordingly the network will be tested with three numbers of neurons; 11, 12 and 13.

Thus, we concerned on five different architectures when the number of hidden layers was two, 25-11-5-1, 25-11-6-1, 25-12-6-1, 25-13-6-1, 25-13-7-1

i. When the network architecture was 25-11-5-1, the number of neurons in the second hidden layer will be 5, therefore number of neurons in the third hidden layer is 5/2=2.5. Since the result is not a fix number applying the floor operation will produce 2, applying the ceil operation will produce 3, then the result is two architectures, 25-11-5-2-1 and 25-11-5-3-1.

ii. When the network architecture was 25-11-6-1, the number of neurons in the second hidden layer will be 6, therefore number of neurons in the third hidden layer is 6/2=3. The network architecture will be in this case 25-11-6-3-1.

iii. When the network architecture was 25-12-6-1, the number of neurons in the second hidden layer will be 6, therefore number of neurons in the third hidden layer is 6/2=3. The network architecture will be in this case 25-12-6-3-1.

iv. When the network architecture was 25-13-6-1, the number of neurons in the second hidden layer will be 6, therefore number of neurons in the third hidden layer is 6/2=3. The network architecture will be in this case 25-13-6-3-1.

v. When the network architecture was 25-13-7-1, the number of neurons in the second hidden layer will be 7, therefore number of neurons in the third hidden layer is 7/2=3.5, and since the result is not a fix number applying the floor operation will produce 3, applying the ceil operation will produce 4, then the result is two architectures, 25-13-7-3-1 and 25-13-7-4-1.

Table (4.3) illustrates the classification accuracy when using three hidden layers; the highest accuracy was achieved with the network architecture of the form 25-13-6-3-1 is 83.3333% classification accuracy. The other architectures achieved results varying in an interval [76.6667-82.8571].

Table 4.3: Classification Accuracy with three hidden layers

Network Architecture	Classification Accuracy
25-11-5-2-1	80.4762
25-11-5-3-1	81.4286
25-11-6-3-1	80
25-12-6-3-1	82.8571
25-13-6-3-1	83.3333
25-13-7-3-1	76.6667
25-13-7-4-1	82.8571

4.2.4 Models' Building using Four Hidden Layers

Building the network with four hidden layers - the highest number of hidden layer that can be achieved- we needs to use the architectures that were obtained using three hidden layers

Table (4.4) illustrates the results obtained using four hidden layers the highest accuracy was achieved with the network architecture of the form 25-11-5-3-2-1 is 85.2381% classification accuracy. The other architectures achieved results varying in an interval [67.619-84.7619].

Table 4.4: Classification Accuracy with four hidden layers

Network Architecture	Classification Accuracy
25-11-5-2-1-1	78.0952
25-11-5-3-1-1	67.619
25-11-5-3-2-1	85.2381
25-11-6-3-1-1	70
25-11-6-3-2-1	84.7619
25-12-6-3-1-1	82.8571
25-12-6-3-2-1	70.9524
25-13-6-3-1-1	76.6667
25-13-6-3-2-1	82.381
25-13-7-3-1-1	82.381
25-13-7-3-2-1	82.381
25-13-7-4-2-1	80.4762

4.2.5 Summary of the results

To summarize, we noticed from the results of all models that:

1. The best classification accuracy was *88.57%* using one hidden layers and 11 hidden neurons.
2. We conclude that experiments are the only way to determine what is the best network architecture that can be used to solve a specific problem.

4.3 System Result and Discussion

The effectiveness of the chosen neural network architecture was evaluated on the diabetes test set for each fold independently. After that, the average accuracy is computed to determine how well the chosen architecture will generalize for new samples that it never seen before. The following subsection will introduce the results for each test set of the 10 folds (runs), where the experimental results of the tenth test sets have also been presented as a confusion matrix only for the ANN model with the highest classification accuracy which is (25-11-1); a confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such a system is commonly evaluated using the data in the matrix. In a confusion matrix, columns represent the predicted data, while rows represent the actual data (Yan et al., 2006).

4.3.1 First Run Result

Table (4.5) illustrated the confusion matrix of the first run. The accuracy for the one oral medication is ((7/7)*100%) which is 100%, for the dual medication it is ((6/7) * 100%) which is 85.71%, and for the insulin medication it is ((7/7) *100%) is 100%. This gives a total accuracy for the first run by applying the equation in section 4.1; (A $_{FI}$ = (20/21)* 100%) = 95.24%. We can notice from the confusion matrix that 14% from the samples of the dual medication were misclassified.

Table 4.5: Confusion Matrix of the First Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	7	0	0	100.00
Dual oral medication	0	6	1	85.71
Insulin medication	0	0	7	100.00
		Total Accuracy		
		95.24		

4.3.2 Second Run Result

Table (4.6) illustrated the confusion matrix of the second run. The accuracy for the one oral medication is ((7/7)*100%) which is 100%, for the dual medication it is ((5/7) * 100%) which is 71.43%, and for the insulin medication it is ((7/7) *100%) is 100%. This gives a total accuracy for the second run (A_{F2} = (19/21)* 100%) = 90.48%. We can notice from the confusion matrix that 29% from the samples of the dual medication were misclassified.

Table 4.6: Confusion Matrix of the Second Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	7	0	0	100.00
Dual oral medication	2	5	0	71.43
Insulin medication	0	0	7	100.00
		Total Accuracy		
		90.48		

4.3.3 Third Run Result

Table (4.7) illustrated the confusion matrix of the third run. The accuracy for the one oral medication is ((7/7)*100%) which is 100%, for the dual medication it is ((4/7) * 100%) which is 57.14%, and for the insulin medication it is ((7/7) *100%) is 100%. This gives a total accuracy for the third run (A_{F3} = (18/21)* 100%) = 85.71%. A percentage of 42% samples from the dual medication were misclassified.

Table 4.7: Confusion Matrix of the Third Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	7	0	0	100.00
Dual oral medication	2	4	1	57.14
Insulin medication	0	0	7	100.00
		Total Accuracy		
		85.71		

4.3.4 Fourth Run Result

Table (4.8) illustrated the confusion matrix of the fourth run. The accuracy for the one oral medication is ((5/7)*100%) which is 71.43%, for the dual medication it is ((6/7) * 100%) which is 85.71%, and for the insulin medication it is ((5/7) *100%) is 71.43%. This gives a total accuracy for the fourth run (A_{F4} = (16/21)* 100%) = 76.19%. We can notice from the confusion matrix that A percentage of 29%, 29% and 20% from the samples of the single medication, dual medication and insulin medication respectively were misclassified

Table 4. 8: Confusion Matrix of the Fourth Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	5	2	0	71.43
Dual oral medication	0	6	1	85.71
Insulin medication	0	2	5	71.43
		Total Accuracy		
		76.19		

4.3.5 Fifth Run Result

Table (4.9) illustrated the confusion matrix of the fifth run. The accuracy for the one oral medication is ((6/7)*100%) which is 85.71%, for the dual medication it is ((6/7) * 100%) which is 85.71%, and for the insulin medication it is ((6/7) *100%) is 85.71%. This gives a total accuracy for the fifth run (A_{F5} = (16/21)* 100%) = 85.71%. We can notice from the confusion matrix that a percentage of 14% from the samples of the single medication, dual medication and insulin medication were misclassified

Table 4. 49: Confusion Matrix of the Fifth Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	6	1	0	85.71
Dual oral medication	0	6	1	85.71
Insulin medication	0	1	6	85.71
		Total Accuracy		
		85.71		

4.3.6 Sixth Run Result

Table (4.10) illustrated the confusion matrix of the sixth run. The accuracy for the one oral medication is ((7/7)*100%) which is 100%, for the dual medication it is ((5/7) * 100%) which is 71.43%, and for the insulin medication it is ((7/7) *100%) is 100%. This gives a total accuracy for the sixth run (A_{F6} = (18/21)* 100%) = 90.48%. the result showed that 29% from the samples of the dual medication were misclassified.

Table 4.10: Confusion Matrix of the Sixth Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	7	0	0	100.00
Dual oral medication	2	5	0	71.43
Insulin medication	0	0	7	100.00
		Total Accuracy		
		90.48		

4.3.7 Seventh Run Result

Table (4.11) illustrated the confusion matrix of the seventh run. The accuracy for the one oral medication is ((7/7)*100%) which is 100%, for the dual medication it is ((6/7) * 100%) which is 85.71%, and for the insulin medication it is ((7/7) *100%) is 100%. This gives a total accuracy for the seventh run (A_{F7} = (18/21)* 100%) = 95.24%..A percentage of 14% from the samples of the dual medication were misclassified.

Table 4. 411: Confusion Matrix of the Seventh Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	7	0	0	100.00
Dual oral medication	1	6	0	85.71
Insulin medication	0	0	7	100.00
		Total Accuracy		
		95.24		

4.3.8 Eighth Run Result

Table (4.12) illustrated the confusion matrix of the eighth run. The accuracy for the one oral medication is ((7/7)*100%) which is 100%, for the dual medication it is ((6/7) * 100%) which is 85.71%, and for the insulin medication it is ((7/7) *100%) is 100%. This gives a total accuracy for the eighth run (A_{F8} = (18/21)* 100%) = 95.24%. We can notice from the confusion matrix that 14% from the samples of the dual medication were misclassified.

Table 4. 12: Confusion Matrix of the Eighth Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	7	0	0	100.00
Dual oral medication	1	6	0	85.71
Insulin medication	0	0	7	100.00
		Total Accuracy		
		95.24		

4.3.9 Ninth Run Result

Table (4.13) illustrated the confusion matrix of the ninth run. The accuracy for the one oral medication is ((6/7)*100%) which is 85.71%, for the dual medication it is ((5/7) *

100%) which is 71.43%, and for the insulin medication it is ((7/7) *100%) is 100%. This gives a total accuracy for the ninth run (A_{F9}= (16/21)* 100%) = 85.71%. We can notice from the confusion matrix that a percentage of 14% and 29% from the samples of the single medication, dual medication respectively were misclassified

Table 4. 13: Confusion Matrix of the Ninth Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	6	1	0	85.71
Dual oral medication	1	5	1	71.43
Insulin medication	0	0	7	100.00
		Total Accuracy		
		85.71		

4.3.10 Tenth Run Result

Table (4.14) illustrated the confusion matrix of the tenth run. The accuracy for the one oral medication is ((7/7)*100%) which is 100%, for the dual medication it is ((6/7) * 100%) which is 85.71%, and for the insulin medication it is ((5/7) *100%) is 71.43%. This gives a total accuracy for the tenth run (A_{F10} = (16/21)* 100%) = 85.71%. We can notice from the confusion matrix that a percentage of 14% and 29% from the samples of the dual medication and insulin medication respectively were misclassified.

Table 4. 14: Confusion Matrix of the Tenth Run

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Classification Accuracy
One oral medication	7	0	0	100.00
Dual oral medication	0	6	1	85.71
Insulin medication	0	2	5	71.43
		Total Accuracy		
		85.71		

4.3.11 Summary of the Results

To summarize the results of the testing results of the ten folds, table (4.15) illustrates the classification accuracy for all results. As we notice the highest classification accuracy obtained in folds 1, 7 and 8 with accuracy of 95.2381% and the lowest accuracy obtained was in fold 4 with accuracy of 76.1905%. Also the total classification accuracy was 88.5714%.

Table 4. 15: Summary of All Classification Accuracy for All Folds

Test Sets	Classification Accuracy %
Fold 1	95.2381
Fold 2	90.4762
Fold 3	85.7143
Fold 4	76.1905
Fold 5	85.7143
Fold 6	90.4762
Fold 7	95.2381
Fold 8	95.2381
Fold 9	85.7143
Fold 10	85.7143
Average Accuracy	88.5714

Table (4.16) illustrates the experimental results of the 10 test sets have also been presented as a confusion matrix.

Table 4.16: The confusion matrix of the therapeutic result for Diabetes: 10- fold cross validation

Regimen Type	One oral medication	Dual oral medication	Insulin medication	Row Sum
One oral medication	66	4	0	70
Dual oral medication	9	55	6	70
Insulin medication	0	5	65	70
Column Sum	75	64	71	210

Form the confusion matrix shown in table (4.16). the accuracy for each class can be calculated, which shows the proportion of the treatment type in the test dataset that have been correctly recognized by ANN, is presented as follows in table (4.17). We notice that the dual medication got the lowest classification accuracy with a percentage of 78.57%.the highest classification accuracy was for the insulin medication with a percentage of 96.86%.

Table 4.17: Proportion of correctly classified Treatment

Treatment type	Accuracy %
One oral medication:	66/70 = 94.29%
Dual oral medication:	55/70 = 78.57%
Insulin medication:	65/70 = 96.86%
Total Accuracy	**186/210 = 88.57%**

Chapter five

5. Conclusion and Future Works

The total number of diabetes patients in the world is expected to grow from 246 Million in 2006 to 380 Million by 2025. Therefore an accurate treatment will play a major role in the life's saving of patients and keep them away from diabetes complication. Beside that many researchers developed clinical decision support system to assist physicians in the diagnosis/treatment process for multiple diseases, where these systems proofed its effectiveness in obtaining high classification accuracy.

Based on that, in this thesis we have developed a clinical decision support system for regimen selection of therapy for diabetes type two patients, which is the first application on that domain. Three used treatment regimens have been considered (single oral medication "***metformin***",dual oral medication "***Metformin Plus Sulfonylurea***", insulin medication) involving twenty five essential clinical variables along with other patient-specific variable for diabetes treatment regimens selection. The proposed system was developed using one of the most widespread ML techniques; a MLP feed forward neural network and trained using the Rprop training algorithm. It consisted of: an input layer with 25 neurons (representing the input variables of each patient such as, age, previous status, weight, etc.), one hidden layer (where the number of neurons equal 11) and an output layer (that produced the type of the treatment). Moreover, a ten-fold cross validation was used to access the generalization of the proposed system using 228 patients' medical recodes collected from the diabetes clinics at JUH. An accuracy of 94.29%, 78.57% and 96.86% were obtained for the one oral medication, dual oral medication and insulin medication.

From these results we conclude that; the proposed system proofed its usefulness in the support of treatment decision, if more dataset was used for training the proposed system that would give more robust results. Moreover, some limitations were noticed in this thesis such as: some medical records contained missing values because they were filled with resident doctors, which made it impossible to use these records in the diabetes disease dataset.

In future, this work can be extended by using other types of ANNs such as, GRNN, PNN and RBF to compare the results obtained from them with the one obtained using the MLP. Also, we can apply one of the feature selection algorithms on the diabetes disease dataset in order to choose the medical variables that play a major role in the treatment process which may give more accurate results. Moreover, applying other ML methods such as Artificial Immune Systems, Neuro-Fuzzy and Decision Trees and comparing the results obtained with the one obtained using the ANN. Finally, since this system was built using the medical records that were collected from JUH which is a training hospital, we can apply the system on the medical records collected from other types of hospitals such as military, public and private hospitals in order to compare the results obtained with the one obtained using JUH's medical records.

REFRENCES

Abu Zitar .R.and Al-Jabali.A.(2005), Towards Neural Network Model for Insulin / Glucose in Diabetics-II, **Informatica 29 227–232**

AlTimemy.A. and Al Naima.F.(2010), Comparison of Different Neural Network Approaches for the Prediction of Kidney Dysfunction, **International Journal of Biological and Life Sciences,** 6 (2), 84-90.

American Diabetes Association, editor (2002). **American Diabetes Association Complete Guide To Diabetes**. Bantam Books, New York, 3rd edition.

Bascil.S.,and Temurtas.F.(2009), A Study on Hepatitis Disease Diagnosis Using Multilayer Neural Network with Levenberg Marquardt Training Algorithm, **Journal of Medical Systems,** 33 (1), 1-4

Brause Rüdiger W. (2001), Medical Analysis and Diagnosis by Neural Networks, **In Proceedings of the Second International Symposium on Medical Data Analysis** London, UK, 1-13.

Coiera E.(2003), **A Guide to Health Informatics.** 2nd edition, Hodder Arnold

Duda, R., Hart, P. and Stork, D. (2001), **Pattern Classification**, (2nd ed) , NewYork: John Wiley & Sons, Inc.

Fausett, L Laurene (2001),**Fundamentals of Neural Networks**, **Architectures, Algorithms and Application**,(1st ed), New Jersey: Pearson Prentice Hall Education , Inc.

Freeman , J and Skapura , D ,(1991) , **Neural Networks Algorithms, Applications, and Programming Techniques,** (1st ed), New York: Addison-Wesley Publishing Company, Inc.

Gan D, editor. Diabetes atlas, 2nd ed. Brussels: **International Diabetes Federation; 2003**. http://www.eatlas.idf.org/webdata/docs/Atlas%202003-Summary.pdf

Geoffrey, H. and Terrence, S. (1999) **Unsupervised Learning: Foundations of Neural Computation** (1st ed), Massachusetts: MIT Press**.**

Ghahramani, Saeed (2000). **Fundamentals of Probability,** (2nd ed). New Jersey: Prentice Hall

Gogou.G., Maglaveras.N., Ambrosiadou.B., Goulis.D., Pappas.C.(2001), A Neural Network Approach in Diabetes Management by Insulin Administration, **journal of Medical Systems**, Vol. 25, No. 2.

Guthrie RA, Guthrie DW(2002), **Nursing management of diabetes mellitus**. 5th ed., New York: Springer Publishing.

Hagan, M, Dcmuth, H, Beale, M.(1996), **Neural Network Design,** (1st ed), USA: PWS Publishing Company.

Haykin, Simon. (2001), **Neural Networks a Comprehensive Foundation,** (2nd ed), India: Pearson Prentice Hall Education , Inc.

Kayaer.K. and Yildirim.T.(2003), Medical Diagnosis on Pima Indian Diabetes Using General Regression Neural Networks**, In Proceedings of the International Conference on Artificial Neural Networks and Neural Information Processing**, Istanbul, turkey,181-184

Kiyan, T. and Yildirim, T. (2003), Breast Cancer Diagnosis Using Statistical Neural Networks, **In International XII Turkish Symposium on Artificial Intelligence and Neural Networks**, Tainn, Turkey, 754-760.

Kong, G., Xu, D. and Yang, J. (2008), Clinical Decision Support Systems: A Review on Knowledge Representation and Inference Under Uncertainties, **International Journal of Computational Intelligence Systems,** 1(2), 159-167.

Krose, B and Smagt, P. (1996), **An introduction To Neural Networks**, (8th ed), Amsterdam: The University of Amsterdam.

Machado, Lucila Ohno.(1996),**Medical Application of Artificial Neural Networks: Connectionist Models of Survival**, Master's thesis, Department of Computer Science, University of Stanford, USA.

Manda R., Rao.N, Sridhar.GR., Madhu.K., Rao.A., 2010, A clinical decision support system using multi-layer perceptron neural network to predict quality of life in diabetes, ***Diabetes and Metabolic Syndrome: Clinical Research and Reviews;*** *57–59*

Mehrotra, K , Chilukuri , K, Mohan and Ranka, S .(1996) ,**Elements of Artificial Neural**

Networks,(1st ed), New York: Addison-Wesley Publishing Company, Inc.

Menéndez, L, Juez , F, Lasheras, F, Riesgo ,J.(2010), Artificial neural networks applied to cancer detection in a breast screening programme , **Mathematical and Computer Modelling**, 52(7), 983-991.

Miller R. and Geissbuhler A. (2007), Diagnostic Decision Support Systems In: Berner E. (Ed), **Clinical Decision Support Systems,** (2nd ed), (pp 99-125), New- York:Springer.

Minino, A., Xu, J.,. and Kochanek, K. (2010), Deaths: Preliminary Data for 2008 (PDF). National Vital Statistics Reports (United States: Center for Disease Control). 59 (2), Retrieved February, 25, 2011 from **http://www.cdc.gov/nchs/products/nvsr.htm**

Mitchell, Tom (1997), **MachineLearning**, (1st ed), McGraw-Hill Science.

Musen Mark A. (1997), Patient Record. In: J.van Bemmel and Mark A. Musen (Ed), **Handbook of medical informatics**, (pp 99 115), Houten: Springer.

Paulin, F. and Santhakumaran, A. (2011), Classification of Breast Cancer by Comparing Backpropagation Training Algorithms, **International Journal on Computer Science and Engineering**, 3(1), 327-332.

Polat.K. and Gunes.S. (2007), An expert system approach based on principal component analysis and adaptive neurofuzzy inference system to diagnosis of diabetes disease, **Digital Signal Processing**, vol. 17, pp. 702-710.

Ramalingam, V., Palaniappan, B., Panchanatham, N. and Palanivel, S. (2006), Measuring Advertisement Effectiveness – A Neural Network Approach, **Expert Systems with Applications,** 31(1), 159–163.

Rasheed, Zeehasham (2009), **Adaptive Fuzzy Logic Based Framework for Handling Imprecision and Uncertainty in Pattern Classification of Bioinformatics Datasets**, Master's thesis, Department of Computer Science, King Fahd University of Petroleum and Minerals, Saudi Arabia.

Riedmiller, M. and Braun ,H. (1993), A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, **Neural Networks, IEEE International Conference, San Francisco, CA , USA** ,586-591.

Temurtas.H., Yumusak.N., Temurtas.F.(2009),A Comparative Study on Diabetes Disease Diagnosis Using Neural Networks, **Expert Systems with Applications,** 36(4), 8610-8615.

Thammano, A., Moolwong, J.(2009),A New Computational Intelligence Technique Based on Human Group Formation, **Expert Systems with Applications**, 37(2), 1628–1634

Thornett, Andrew M. (2001), Computer Decision Support Systems in General Practice, **International Journal of Information Management,** 21(1), 39-47.

Vahdani.P., Alavian.S., Aminzadeh.Z., Raoufy.M., Gharibzadeh.S., Vahdani.G., Fekri.S., and Eftekhari.P.(2009), Using Artificial Neural Network to Predict Cirrhosis in Patients with Chronic Hepatitis B Infection with Seven Routine Laboratory Findings, **Hepatitis Monthly ,** 9(4), 271-275

Veropoulos, Konstantinos (2001), **Machine Learning Approaches to Medical decision Making,** Doctoral thesis, Department of Engineering Mathematics, University of Bristol, United Kingdom.

Xu,S. and Chen, L. (2008), A Novel Approach for Determining the Optimal Number of Hidden Layer Neurons for FNN's and Its Application in Data Mining**, In the 5th International Conference on Information Technology and Applications,** Cairns, Australia, 683-686

Yan, H., Jiang, Y., Zheng, J., Peng, C. and Li, Q. (2006), A Multilayer Perceptron-Based Medical Decision Support System for Heart Disease Diagnosis, **Expert Systems with Applications**, 30(2), 272-281.

Zainuddin.Z., Pauline.O., Ardil.C.(2009), *A Neural Network Approach in Predicting the Blood Glucose Level for Diabetic Patients,* **International Journal of Information and Mathematical Sciences.**

نظام لدعم القرار الطبي معالجة مرض السكري

إعداد
تسنيم محمود محمد ابو كبير

المشرف
الدكتور صالح الشرايعه

المشرف المشارك
الدكتوره فريال الهياجنه

ملخص
يعتبر مرض السكري من أكثر الأمراض انتشارا بين الناس ذلك حسب منظمة الصحة العالمية إذ أن هنالك 194 مليون شخص في العالم مصاب بمرض السكري والتوقعات تشير بازدياد عدد المصابين إلى 300 مليون مريض بحلول عام 2050 , تكمن خطورته في مضاعفاته الكثيرة التي تدمر أجهزة الجسم المختلفة وتجعلها بحالة عجز وشلل, هذا لكونه مرض مزمن لا يوجد علاج ناجع له, لكن هناك وسائل كثيرة ناجحة وفعالة للسيطرة عليه والحد من مضاعفاته ومن أهمها العلاج بالأدوية اذ يوجد نوعين من العلاج , الأول يؤخذ عن طريق الفم ويعمل على تحفيز البنكرياس لإفراز الأنسولين , أما النوع الثاني ، فهي حقن الأنسولين التي تصب مباشرة في مجرى الدم وتقوم بخفض نسبة السكر فورا.

لقد اتجه الكثير من الباحثين في العقود الأخيرة إلى تطوير أنظمة لدعم القرارات الطبية، حيث أن الهدف من هذه الأنظمة مساعدة الطبيب في عملية تشخيص و معالجه الأمراض وبالتالي التقليل من نسبة الخطأ الطبي الممكن حدوثه في عملية التشخيص. بناءا على ذلك تم اعتماد الكثير من المنهجيات والوسائل لتحقيق هذا الهدف. أحد أشهر هذه الوسائل والتي تنتمي لعائلة الحسابات الذكية هي: الشبكات العصبية الاصطناعية المستوحاة فكرتها من الأنظمة العصبية البيولوجية.

في هذه الرسالة تم تصميم نظاما لدعم القرار الطبي لإعطاء العلاج المناسب لمرضى السكري بواسطة شبكة عصبية اصطناعية متعددة الطبقات ذات تغذية أمامية. تستقبل طبقة الإدخال 25 متغير تم ترميزهم بالطريقة المقترحة بهذه الرسالة. في حين أن طبقة الإخراج ستنتج نوع العلاج المناسب الذي يقلل من احتماليه إصابته بالمضاعفات .لقد تم استعمال طريقة تكرارية لتحديد عدد الطبقات المخفية و عدد الخلايا العصبية في هذه الطبقات. لتدريب الشبكة تم استعمال خوارزمية التدريب (RProp). بالإضافة إلى ذلك تم تطبيق طريقة المجموعات العشر للتحقق من مقدرة الشبكة العصبية على التعامل مع بيانات جديدة لم تتعامل معها في مرحلة التدريب.
لقد تم تدريب وإجراء الاختبارات على النظام بواسطة 228 سجل طبي تم الحصول عليها من مستشفى الجامعة الأردنية. وقد أظهرت التجارب بأن النظام المقترح في هذه الرسالة استطاع التنبؤ بطريقة معالجة مرضى السكري بدقة 88.57%